

陈军红
王瑞敬 编著

PHP编程
从基础到应用

清华大学出版社



陈军红
王瑞敬 编著

PHP编程
从基础到应用

清华大学出版社
北 京

内 容 简 介

本书从初学者的角度出发，由浅入深、循序渐进地介绍用 PHP 进行 Web 开发必备的知识和技能。主要包括搭建 PHP 开发环境、PHP 数据类型、运算符和表达式、条件和循环语句、PHP 的面向对象编程、操作 PHP 的数组和字符串、生成 XML、获取时间、文件打开与写入、保存页面数据、读取数据库以及 Ajax 等。最后以一个 PHP 和 MySQL 整合的相册管理系统讲解 PHP 在实际 Web 开发中的应用。

本书内容丰富、实例精彩、覆盖面广、指导性强，以全面的知识及丰富的实例来指导读者透彻地学习用 PHP 进行 Web 开发的知识。

图书在版编目（CIP）数据

PHP 编程从基础到应用/陈军红等编著. —北京：清华大学出版社，2014

从基础到应用

ISBN 978-7-302-31794-4

Ⅰ. ①P…　Ⅱ. ①陈…　Ⅲ. ①PHP 语言-程序设计-教材　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2013）第 062984 号

责任编辑：夏兆彦
封面设计：胡文航
责任校对：胡伟民
责任印制：

出版发行：清华大学出版社
　网　　址：http://www.tup.com.cn, http://www.wqbook.com
　地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
　社 总 机：010-62770175　　**邮　　购：**010-62786544
　投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
　质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　**印　张：**29.75　**字　数：**745 千字
附光盘
版　　次：2013 年 8 月第 1 版　**印　次：**2013 年 8 月第 1 次印刷
印　　数：1～4000
定　　价：59.00 元

产品编号：045972-01

FOREWORD

前言

在互联网刚刚兴起时，Web 开发者多使用 C 或者 Perl 等 CGI 语言进行 Web 开发，进而发展到使用 PHP 以及 ASP 等脚本语言。经过十几年的时间，PHP 在 Web 开发领域占有绝对优势，许多互联网公司都在使用 PHP，像百度、新浪、盛大和金山等。而且 PHP 仍然保持着昂扬的斗志在前进，它不断增强升级，每一个版本更新都带给业界和 Web 开发者众多惊喜。

事实与时间均证明，PHP 已经成为全球最受欢迎的 Web 开发语言之一。本书以最新的 PHP 5.4 为例由易到难、由浅入深、循序渐进、详细而系统地介绍使用 PHP 进行 Web 开发的技术。

本书内容

第 1 章：创建第一个 PHP 程序。本章主要介绍 Apache 和 PHP 的安装、集成环境的使用、PHP 的基本语法、PHP 的输出语句以及注释的使用等。

第 2 章：PHP 语法快速入门。本章详细介绍 PHP 的基础语法帮助读者快速入门，包括常量和变量、变量作用域、数据类型、类型检查和转换、各种类型的运算符以及运算符的优先规则等。

第 3 章：PHP 程序流程控制。本章主要介绍程序控制的三类语句，像单分支 if、多分支嵌套、for 循环、do 循环、return 返回，还介绍文件引用语句的使用。

第 4 章：PHP 类编程。本章重点对 PHP 中面向对象的实现进行介绍，包括创建类、构造函数、类常量、类的方法、PHP 作用域关键字以及继承的实现等。

第 5 章：数组处理。本章详细介绍对数组的各种处理，像测试某个变量是否是数组、遍历数组内容、对数组进行计算和查询，以及数组的排序等。

第 6 章：字符串处理。本章详细介绍对字符串的各种操作，像创建多行字符串、获取字符串长度和单词数量、大小写替换、去除多余字符、分隔字符串，以及字符串的填充和替换等。

第 7 章：常用数据处理。本章首先介绍自定义函数的使用和调用，然后介绍处理数学、日期和时间的方法，接下来对 XML 和正则表达的使用进行讲解。

第 8 章：文件和目录处理。本章主要介绍 PHP 中对文件和目录的处理，包括查看文件属性、打开文件、读取一行/指定字节/全部内

容、写入内容、创建和删除目录以及读取目录所在硬盘空间等。

第 9 章：与 Web 页面交互。本章首先讲解表单的知识，然后介绍如何获取表单数据、遍历表单以及处理技巧，接下来介绍 Cookie 和 Session 的使用、文件的上传以及下载等。

第 10 章：MySQL 数据库与 PHP 处理。本章首先介绍 MySQL 数据库的安装，然后重点介绍在 PHP 中使用 mysql 和 mysqli 库对 MySQL 进行操作，像连接数据库、执行 SQL 语句、获取结果集以及显示错误信息等。

第 11 章：PHP 高级开发。本章主要介绍 PHP 下 Ajax 的开发，同时对加密技术和开发规范也有简单介绍。

第 12 章：相册管理系统。本章介绍使用 PHP 实现相册的过程，包括功能分析、数据库设计、搭建框架、浏览相册列表、查看图片详细、创建相册以及上传图片等。

本书特色

本书采用大量的实例进行讲解，力求通过实际操作使读者轻松掌握 PHP 技术。本书每章后面都有精选的习题供读者巩固所学知识，每个章节末尾都有实践疑难解答，在这个模块中收集了每章的重点、难点以及易出错点并以问答的形式呈现给读者，使读者更轻松地掌握这些知识。本书难度适中，内容由浅而深、实用性强、覆盖面广、条理清晰。

- 知识全面　本书全面地介绍 PHP Web 开发的知识点，具有很强的系统性。
- 示例典型，应用广泛　作者精心挑选了大量示例程序，它们都是根据作者在实际开发中的经验总结而来的，涵盖了在实际开发中出现的各种问题，而且有些程序可以直接在项目中使用，无须二次开发。
- 快速掌握　注重技术原理和实际应用的高度融合，通过循序渐进的内容组织，帮助读者快速掌握和应用 PHP 进行 Web 开发。
- 随书光盘　本书为实例配备了视频教学文件，读者可以通过视频文件更加直观地学习 PHP 相关知识。
- 网站技术支持　读者在学习或者工作的过程中，如果遇到实际问题，可以直接登录 www.itzcn.com 与我们取得联系，作者会在第一时间内给予帮助。

读者对象

本书具有知识全面、实例精彩、指导性强的特点，力求以全面的知识及丰富的实例来指导读者透彻地学习 PHP 开发 Web 方面的知识。本书可以作为 PHP 的入门书籍，也可以帮助中级读者提高技能，对高级读者也有一定的启发意义。

本书适合以下人员阅读学习。

- PHP 初学者。
- 其他 Web 开发的从业人员。
- 准备从事 Web 开发的求职者。
- 高等院校计算机相关专业的老师和学生。

除了封面署名人员之外，参与本书编写的人员还有马海军、李海庆、陶丽、王咏梅、康显丽、郝军启、朱俊成、宋强、孙洪叶、袁江涛、张东平、吴鹏、王新伟、刘青凤、汤莉、冀明、王超英、王丹花、闫琰、张丽莉、李卫平、王慧、牛红惠、丁国庆、黄锦刚、李旎、王中行、李志国等。在编写过程中难免会有漏洞，欢迎读者通过我们的网站www.itzcn.com与我们联系，帮助我们改正提高。

CONTENTS

目录

第 1 章　创建第一个 PHP 程序 1
1.1　PHP 简介 1
1.1.1　PHP 历史 1
1.1.2　PHP 特点 2
1.2　全新方式搭建 PHP 环境 3
1.2.1　安装 Apache 3
1.2.2　安装 PHP 5
1.3　集成方式搭建 PHP 环境 8
1.3.1　WampServer 8
1.3.2　PHPnow 10
1.4　查看 PHP 配置文件 12
1.5　选择 PHP 语法风格 13
1.5.1　默认标记 13
1.5.2　ASP 风格标记 14
1.5.3　脚本标记 14
1.5.4　短标记 14
1.6　向页面输出内容 16
1.6.1　输出字符串 16
1.6.2　格式化输出字符串 17
1.7　程序注释 19
1.7.1　单行注释 20
1.7.2　多行注释 20
1.8　项目案例：自定义 Apache 的主目录 21
1.9　项目案例：在 IIS 上配置 PHP 环境 22
1.10　习题 25
1.11　实践疑难解答 26
1.11.1　php.ini 不起作用的问题 26
1.11.2　安装成功，访问 PHP 脚本时出错 27

第 2 章　PHP 语法快速入门 28
2.1　常量 28
2.1.1　声明和使用常量 28
2.1.2　系统常量 30
2.2　变量 30
2.2.1　变量的命名规则 30
2.2.2　变量赋值 31

2.3.3 可变变量 32
2.3.4 系统变量 32
2.3.5 变量作用域 33
2.3 数据类型 36
2.3.1 标量数据类型 36
2.3.2 复合数据类型 38
2.3.3 特殊数据类型 39
2.3.4 类型自动转换 40
2.3.5 类型强制转换 41
2.3.6 与类型有关的函数 42
2.4 运算符 44
2.4.1 赋值运算符 44
2.4.2 字符串运算符 45
2.4.3 算术运算符 45
2.4.4 递增和递减运算符 46
2.4.5 位运算符 47
2.4.6 逻辑运算符 48
2.4.7 比较运算符 49
2.4.8 条件运算符 50
2.4.9 错误控制运算符 50
2.4.10 运算符的优先规则 51
2.5 习题 52
2.6 实践疑难解答 53
2.6.1 条件运算符计算结果的问题 53
2.6.2 关于自增和自减运算的疑问 54
2.6.3 如何求表达式的值 55

第 3 章 PHP 程序流程控制 56
3.1 顺序结构 56
3.1.1 语句编写方式 56
3.1.2 表达式语句 57
3.1.3 空语句 57
3.1.4 复合语句 58
3.2 分支结构 58
3.2.1 单分支 59
3.2.2 双分支 60
3.2.3 多分支 62
3.2.4 分支嵌套 65
3.2.5 多分支的另一种实现 67
3.3 循环结构 70
3.3.1 while 语句 70
3.3.2 do while 语句 72
3.3.3 for 语句 73
3.3.4 foreach 语句 76
3.4 跳转结构 78
3.4.1 return 语句 78
3.4.2 break 语句 79
3.4.3 continue 语句 80
3.5 文件引用语句 81
3.5.1 include 和 include_once 81
3.5.2 require 和 require_once 82
3.6 项目案例：制作一个 PHP 网站首页 85
3.7 习题 88
3.8 实践疑难解答 91
3.8.1 使用 switch 控制范围出现的问题 91
3.8.2 PHP 中 exit、continue 和 break 的解释 92
3.8.3 do while 循环和 while 循环的区别 92

第 4 章 PHP 类编程 93
4.1 面向对象简介 93
4.1.1 对象的概念 93
4.1.2 抽象性 94
4.1.3 封装性 95
4.1.4 继承性 95
4.1.5 多态性 96
4.2 类的基本应用 97
4.2.1 定义类 97
4.2.2 实例化类 97
4.2.3 构造函数 98
4.2.4 析构函数 99
4.3 类的成员 100
4.3.1 常量 100
4.3.2 字段 101
4.3.3 属性 103
4.3.4 方法 105
4.4 作用域关键字 107
4.4.1 abstract 关键字 107
4.4.2 final 关键字 108
4.4.3 private 关键字 109

4.4.4 protected 关键字 …… 110
4.4.5 public 关键字 …… 110
4.4.6 static 关键字 …… 111
4.5 对象继承 …… 113
4.5.1 继承类 …… 113
4.5.2 继承构造函数 …… 114
4.6 项目案例：实现三层架构的用户登录 …… 115
4.7 习题 …… 121
4.8 实践疑难解答 …… 124
4.8.1 PHP 类变量的问题 …… 124
4.8.2 关于 PHP 类的私有属性的引用问题 …… 125

第 5 章 数组处理 …… 127
5.1 创建数组 …… 127
5.1.1 使用赋值创建数组 …… 127
5.1.2 使用 array()函数创建数组 …… 129
5.1.3 创建多维数组 …… 130
5.2 使用数组 …… 130
5.2.1 测试数组 …… 131
5.2.2 输出数组内容 …… 131
5.3 遍历数组 …… 132
5.3.1 foreach 语句遍历 …… 132
5.3.2 for 语句遍历 …… 133
5.3.3 each()函数遍历 …… 134
5.3.4 遍历数组函数 …… 135
5.4 数组计算 …… 136
5.4.1 计算元素总数 …… 136
5.4.2 计算元素出现的频率 …… 137
5.4.3 计算出现的所有元素 …… 138
5.5 数组元素操作 …… 138
5.5.1 增加元素 …… 138
5.5.2 删除元素 …… 141
5.5.3 定位元素 …… 142
5.5.4 提取元素 …… 146
5.6 数组操作 …… 149
5.6.1 截取数组 …… 149
5.6.2 合并数组 …… 152
5.7 数组排序 …… 155
5.7.1 按值排序 …… 155
5.7.2 按键排序 …… 158
5.7.3 关联排序 …… 159
5.7.4 级联排序 …… 161
5.7.5 自定义排序 …… 163
5.8 项目案例：制作查看教程页面 …… 164
5.9 习题 …… 167
5.10 实践疑难解答 …… 170
5.10.1 如何返回数组中相同键值的键名 …… 170
5.10.2 怎样把同一数组中相同的键值合并为一个 …… 171

第 6 章 字符串处理 …… 172
6.1 创建字符串 …… 172
6.1.1 字符串与数组的转换 …… 172
6.1.2 双引号创建 …… 173
6.1.3 单引号创建 …… 174
6.1.4 定界符创建 …… 175
6.2 统计字符串 …… 175
6.2.1 统计字符串长度 …… 176
6.2.2 统计字符出现频率 …… 176
6.2.3 统计单词数量 …… 177
6.3 操作字符串内容 …… 178
6.3.1 大小写替换 …… 178
6.3.2 去除空格和特殊字符 …… 181
6.3.3 比较字符串 …… 183
6.3.4 查找字符串 …… 185
6.4 操作子字符串 …… 187
6.4.1 分隔字符串 …… 187
6.4.2 填充字符串 …… 189
6.4.3 截取字符串 …… 190
6.4.4 替换字符串 …… 193
6.5 习题 …… 195
6.6 实践疑难解答 …… 197
6.6.1 PHP 加法运算中如果包含了字符串是怎么处理的 …… 197
6.6.2 提取 URL 中字符串参数的问题 …… 198

第 7 章 常用数据处理 …… 200
7.1 用户函数 …… 200
7.1.1 函数定义语法结构 …… 200

7.1.2 使用函数 …… 201
7.1.3 函数返回值 …… 203
7.1.4 函数参数传递方式 …… 203
7.1.5 递归函数 …… 207
7.1.6 嵌套函数 …… 208
7.1.7 判断函数是否存在 …… 208
7.2 数学运算 …… 209
7.3 日期和时间运算 …… 211
7.3.1 UNIX 时间戳 …… 211
7.3.2 日期函数 …… 211
7.3.3 时间函数 …… 215
7.4 XML …… 217
7.4.1 了解 XML 的结构 …… 217
7.4.2 创建一个 XML 文档 …… 219
7.4.3 SAX 解析 XML …… 220
7.4.4 DOM 解析 XML …… 223
7.4.5 SimpleXML 解析 XML …… 227
7.5 正则表达式 …… 230
7.5.1 POSIX 正则表达式语法 …… 230
7.5.2 POSIX 正则表达式函数 …… 232
7.5.3 Perl 正则表达式语法 …… 235
7.5.4 Perl 正则表达式函数 …… 237
7.6 项目案例：实现基于 XML 的广告位管理 …… 240
7.7 习题 …… 246
7.8 实践疑难解答 …… 249
7.8.1 使用 date()函数出错的问题 …… 249
7.8.2 SimpleXML 的一点注意事项 …… 250
7.8.3 请教 PHP 正则表达式过滤和替换的问题 …… 250

第 8 章 文件和目录处理 …… 252
8.1 查看文件属性 …… 252
8.1.1 filetype()函数 …… 253
8.1.2 fstat()函数 …… 254
8.2 打开和关闭文件 …… 255
8.2.1 打开文件 …… 255
8.2.2 关闭文件 …… 256
8.3 读取文件 …… 257
8.3.1 读取一行 …… 257
8.3.2 读取指定字节 …… 261
8.3.3 读取全部内容 …… 263
8.3.4 其他读取函数 …… 264
8.4 移动文件指针 …… 265
8.4.1 fseek()函数 …… 265
8.4.2 ftell()函数 …… 266
8.4.3 rewind()函数 …… 266
8.5 写入文件 …… 267
8.5.1 fwrite()函数 …… 267
8.5.2 fputs()函数 …… 269
8.5.3 file_put_contents()函数 …… 270
8.6 操作文件 …… 271
8.6.1 复制文件 …… 271
8.6.2 重命名文件 …… 272
8.6.3 删除文件 …… 272
8.7 操作目录 …… 273
8.7.1 打开目录 …… 273
8.7.2 关闭目录 …… 273
8.7.3 遍历目录 …… 274
8.7.4 创建目录 …… 276
8.7.5 删除目录 …… 277
8.8 解析路径 …… 277
8.8.1 获取文件名 …… 277
8.8.2 获取目录部分 …… 278
8.8.3 获取路径中的各个部分 …… 278
8.8.4 获取绝对路径 …… 279
8.9 读取磁盘属性 …… 280
8.9.1 获取目录所在磁盘的可用空间 …… 280
8.9.2 获取磁盘总容量 …… 280
8.9.3 获取目录占用空间 …… 281
8.10 项目案例：简单文件管理系统 …… 282
8.11 习题 …… 289
8.12 实践疑难解答 …… 291
8.12.1 删除目录及目录下所有文件的问题 …… 291
8.12.2 如何递归遍历一个文件夹下面的层次目录 …… 292

第 9 章 与 Web 页面交互 …… 294
9.1 表单 …… 294
9.1.1 表单与 HTML …… 294

9.1.2 表单与 PHP ······295
9.2 获取表单数据······298
9.2.1 设置表单提交方式······298
9.2.2 获取 GET 提交的数据······299
9.2.3 获取 POST 提交的数据······302
9.3 表单的常见操作······303
9.3.1 遍历表单······303
9.3.2 获取表单中的多值······305
9.3.3 动态生成表单······306
9.4 表单处理技巧······311
9.4.1 检测表单提交路径······311
9.4.2 避免表单重复提交······312
9.4.3 表单过期处理······315
9.5 转换 URL 中的汉字······316
9.5.1 编码操作······316
9.5.2 解码操作······317
9.6 Cookie 存储数据······318
9.6.1 Cookie 概述······318
9.6.2 向 Cookie 保存数据······319
9.6.3 从 Cookie 读取数据······321
9.6.4 删除 Cookie 数据······322
9.7 Session 存储数据······323
9.7.1 Session 概述······324
9.7.2 向 Session 保存数据······324
9.7.3 从 Session 读取数据······326
9.7.4 删除 Session 数据······330
9.7.5 Session 数据的编码和解码······331
9.8 文件上传······333
9.8.1 准备文件上传表单······333
9.8.2 处理上传文件······334
9.9 文件下载······336
9.10 项目案例：制作简单留言本······338
9.11 习题······344
9.12 实践疑难解答······347
9.12.1 关于表单提交的问题······347
9.12.2 表单验证 JavaScript 和 PHP 哪个消耗的数据流量更大······347
9.12.3 session_destroy()的问题······348
9.12.4 文件下载的实现······348

第 10 章 MySQL 数据库与 PHP 处理······351
10.1 MySQL 数据库······351
10.1.1 安装 MySQL 数据库······351
10.1.2 配置 MySQL 数据库······353
10.1.3 基本操作······356
10.2 PHP 连接 MySQL 方式······358
10.2.1 mysql 库······359
10.2.2 mysqli 库······359
10.3 连接 MySQL 数据库······360
10.3.1 建立连接······360
10.3.2 关闭连接······363
10.3.3 选择数据库······363
10.4 基本操作······364
10.4.1 获取结果集······364
10.4.2 显示结果集······369
10.4.3 执行 SQL 语句······373
10.5 辅助函数······376
10.6 显示 MySQL 数据库信息······377
10.6.1 获取错误信息······378
10.6.2 获取数据库信息······379
10.6.3 获取数据表信息······381
10.6.4 获取列信息······382
10.7 使用 mysqli······387
10.7.1 基本操作······387
10.7.2 获取结果集······388
10.7.3 使用预处理语句······390
10.8 项目案例：实现基于数据库的留言本······393
10.9 习题······399
10.10 实践疑难解答······402
10.10.1 PHP+MySQL 文字乱码显示问题······402
10.10.2 缺少 mysqli 扩展的问题······402

第 11 章 PHP 高级开发······404
11.1 使用 Ajax 异步通信······404
11.1.1 Ajax 简介······404
11.1.2 XMLHttpRequest 对象简介······405
11.1.3 处理文本······408

11.1.4 处理 XML ······411
11.2 PHP 加密技术 ······413
11.2.1 内置加密函数 ······413
11.2.2 加密扩展 ······415
11.3 PHP 开发编程规范 ······418
11.3.1 包含文件 ······419
11.3.2 命名规范 ······419
11.3.3 代码编写规范 ······421
11.3.4 程序注释 ······424
11.3.5 项目结构规范 ······425
11.4 习题 ······426
11.5 实践疑难解答 ······428
11.5.1 如何解决 PHP 接收的参数是乱码问题 ······428
11.5.2 关于会员注册时密码加密的问题 ······429

第 12 章 相册管理系统 ······431

12.1 系统分析 ······431
12.1.1 功能分析 ······431
12.1.2 数据库设计 ······432
12.2 公共模块 ······433
12.2.1 搭建项目架构 ······434
12.2.2 设计通用类 ······434
12.2.3 设计类库 ······437
12.2.4 设计模型类 ······438
12.2.5 配置文件 ······439
12.3 前台功能实现 ······440
12.3.1 查看所有相册 ······440
12.3.2 查看相册图片 ······443
12.3.3 查看图片详情 ······446
12.3.4 随便看看 ······448
12.4 管理员登录 ······450
12.5 后台功能实现 ······452
12.5.1 创建相册 ······452
12.5.2 上传图片 ······454
12.5.3 图片管理 ······455
12.5.4 相册管理 ······457

参考答案 ······459

第 1 章 创建第一个 PHP 程序 ······459
第 2 章 PHP 语法快速入门 ······459
第 3 章 PHP 程序流程控制 ······459
第 4 章 PHP 类编程 ······460
第 5 章 数组处理 ······460
第 6 章 字符串处理 ······461
第 7 章 常用数据处理 ······461
第 8 章 文件和目录处理 ······462
第 9 章 与 Web 页面交互 ······462
第 10 章 MySQL 数据库与 PHP 处理 ······463
第 11 章 PHP 高级开发 ······463

第 1 章 创建第一个 PHP 程序

PHP 是目前流行的动态网页开发语言之一，因为 PHP 是一种易于学习和使用的服务器端脚本语言。用户只需要具备很少的编程知识，就可以使用 PHP 建立一个具有交互功能的 Web 站点。PHP 是一种嵌入式 HTML 脚本语言，大多数语法来源于 C，也有一部分 PHP 特性借鉴于 Java 和 Perl。这种语言的目的是让 Web 开发人员能够快速高效地写出动态生成的页面。

本章首先向读者介绍 PHP 的背景知识，然后详细介绍 PHP 环境的搭建方法以及集成环境的使用。接下来讲解一些 PHP 的入门语法，像语法风格、输出函数以及注释的使用等。

本章学习要点：

- 了解 PHP 的历史和特点
- 掌握 Apache 的安装和测试方法
- 掌握 PHP 的安装和配置方法
- 掌握 WampServer 和 PHPnow 的安装
- 熟悉 PHP 配置文件的结构
- 掌握使用 PHP 的 4 种语法
- 掌握常用的 PHP 输出函数
- 掌握 PHP 注释的使用

1.1 PHP 简介

构建一个动态网站，有多种动态网站开发技术可以选择，如 JSP、PHP、ASP 等。在众多的动态网站开发技术中，PHP 是一种易于学习和使用的后台开发技术。

下面首先从 PHP 的历史开始介绍，为后面深入学习 PHP 知识打下基础。

1.1.1 PHP 历史

PHP 最初为 Personal Home Page 的缩写，表示个人主页，于 1994 年由 Rasmus Lerdorf 创建。这些工具程序最初用来显示 Rasmus Lerdorf 的个人履历，以及统计网页流量。后来又用 C 语言重新编写，包括可以访问数据库。他将这些程序和一些表单编译器整合起来，称为 PHP/FI。

在 1995 年早期发布了 PHP 的第一个版本 PHP 1.0。在这一版本中，提供了访客留言本、

访客计数器等简单的功能。随着越来越多的人使用 PHP 1.0，1995 年 6 月 8 日作者将源代码公开发布，希望可以通过社群来加速程序开发与寻找错误。这个发布的版本命名为 PHP 2。

PHP 2 已经具有今日 PHP 的一些雏形，像是类似 Perl 的变量命名方式、表单处理功能以及嵌入到 HTML 中执行的能力。程序语法上也类似 Perl，有较多的限制，不过更简单、更有弹性。PHP/FI 还加入了对 MySQL 的支持，从此建立了 PHP 在动态网页开发上的地位。

1997 年重新编写了之前的解释器，并在 1998 年 6 月正式发布 PHP 3 版本，同时 PHP 也改为 Hypertext Preprocessor 的首字母缩写。

1999 年，PHP 的核心解释器被重命名为 Zend Engine（Zend 引擎）。PHP 引擎会将用户经常访问的 PHP 程序驻留在内存中。当其他用户再次访问这个程序时不再重新编译程序，直接执行内存中的代码，这也是 PHP 高效率的体现之一。图 1-1 所示为 PHP 的运行机制。

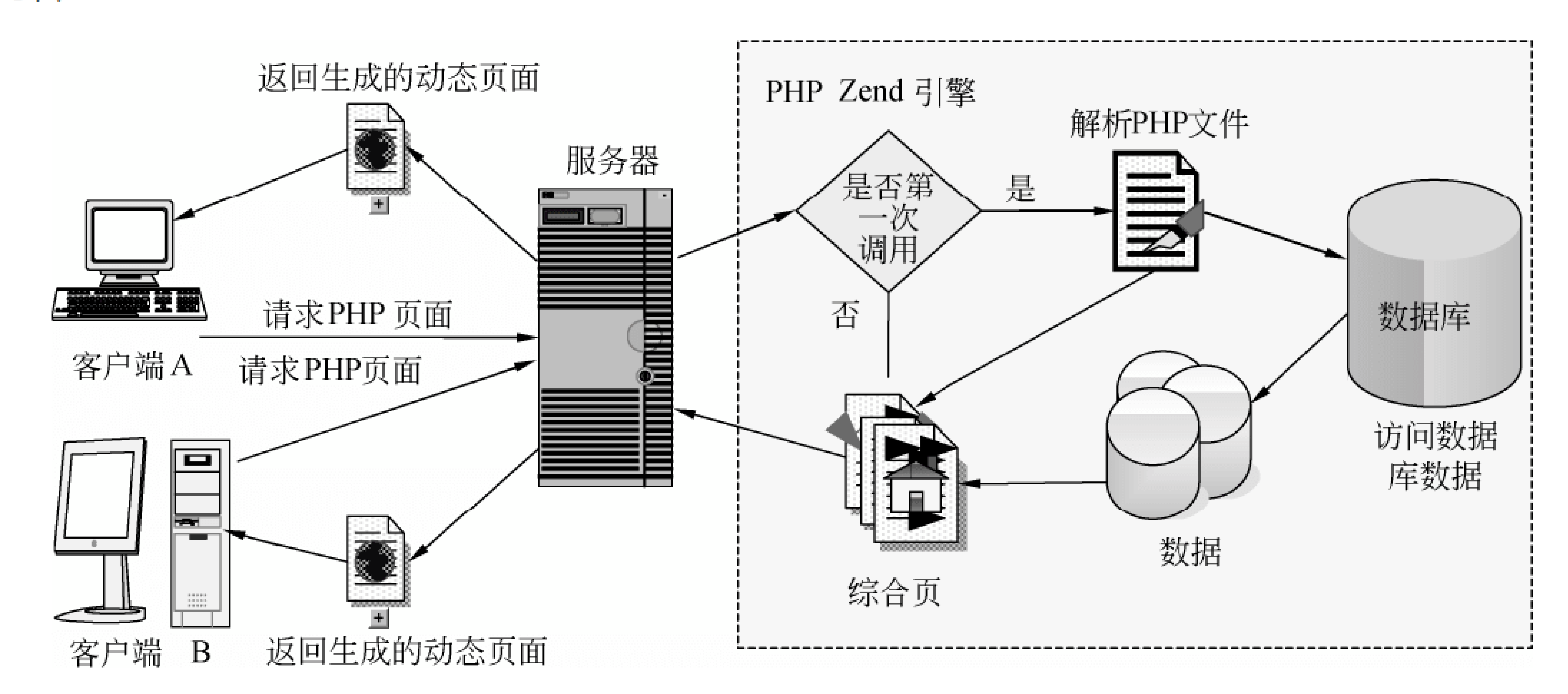


图 1-1　PHP 运行机制

2000 年 5 月 22 日，以 Zend Engine 1.0 为基础的 PHP 4 正式发布。2004 年 7 月 13 日则发布了 PHP 5，PHP 5 使用了第二代的 Zend Engine。PHP 包含了许多新特色，像强化的面向对象功能、引入 PDO（PHP Data Objects）以及许多性能上的增强。目前 PHP 4 已经不会继续更新，以鼓励用户转移到 PHP 5。

2008 年，PHP 5 成为了 PHP 唯一的有再开发的 PHP 版本。PHP 5.3 加入了 Late static binding 和一些其他的功能强化。PHP 6 的开发也正在进行中，主要的改进有移除 register_globals、magic quotes 和 Safe mode 的功能。

有关 PHP 的更多内容可以访问网址 http://www.php.net。

1.1.2　PHP 特点

PHP 一经出现便得到广泛的使用，并受到开发人员的喜爱，主要是因为 PHP 具有如下特点。

- **开放源代码　所有的 PHP 源代码都可以得到。**

- ❑ 免费　与其他技术相比，PHP 是免费的。
- ❑ **PHP 的快捷性**　程序开发快、运行快、入门快。
- ❑ 嵌入于 **HTML**　因为 PHP 可以被嵌入于 HTML 语言，与其他语言相比，PHP 编辑简单、实用性强，更适合初学者。
- ❑ 跨平台性强　由于 PHP 是运行在服务器端的脚本，可以运行在 UNIX、Linux 或者 Windows 下。
- ❑ 效率高　PHP 消耗相当少的系统资源。
- ❑ 图像处理　用 PHP 可动态创建图像。
- ❑ 面向对象　从 PHP 4 起面向对象方面有了很大的改进，现在 PHP 完全可以用来开发大型商业程序。
- ❑ 专业专注　PHP 支持脚本语言为主，同为类 C 语言。

目前 PHP 的最新版本是 PHP 5.4，这是自 PHP 5.3 后的又一次主版本升级。此次升级改动较为显著，删除了一些过时的函数，带来了高达 20%的速度提升和更少的内存使用。

此次更新的关键特性包括：新增 traits，更精简的 Array 数组语法，供测试使用的内建 webserver，可以闭包使用的$this 指针，实例化类成员访问。

同时 PHP 5.4 对性能有大幅提升，修复超过 100 个 bug。废除了 register_globals、magic_quotes 以及安全模式。另外值得一提的是多字节支持已经默认启用了，default_charset 从 ISO-8859-1 已经变为 UTF-8，默认发送“Content-Type: text/html; charset=utf-8”。因此开发人员再也不需要在 HTML 里写 meta 标记，也无须为 UTF-8 兼容而传送额外的 header 了。

1.2　全新方式搭建 PHP 环境

通过上节的介绍，读者应该对 PHP 有了理论上的认识。这一节，读者将自己动手在本地计算机搭建 PHP 的运行环境。由于 PHP 具有跨平台特性，它可以运行在 Linux、Mac OS、UNIX 和 Windows 等操作系统上。各个系统上的搭建方式也不相同，这里以 Windows 操作系统中的搭建为例进行介绍。

在 Windows 下可以使用全新方式和集成方式两种方法来搭建 PHP 环境。全新方式是指手动安装运行 PHP 所需的 Web 服务器和 PHP 的运行库，而集成方式则通过软件来自动搭建。

首先介绍全新方式的搭建过程，这种方式有助于读者更好地理解 PHP 的运行过程。

1.2.1　安装 Apache

Apache 是目前使用最多的 Web 服务器，而且版本更新速度非常快，并且支持多种操作系统。因此，在安装 Apache 之前应该下载与当前操作系统匹配的版本。

Apache 官方网站地址为 http://httpd.apache.org，图 1-2 所示为首页运行效果，从中可以看到目前 Apache 的最新版本为 Apache 2.4.2。



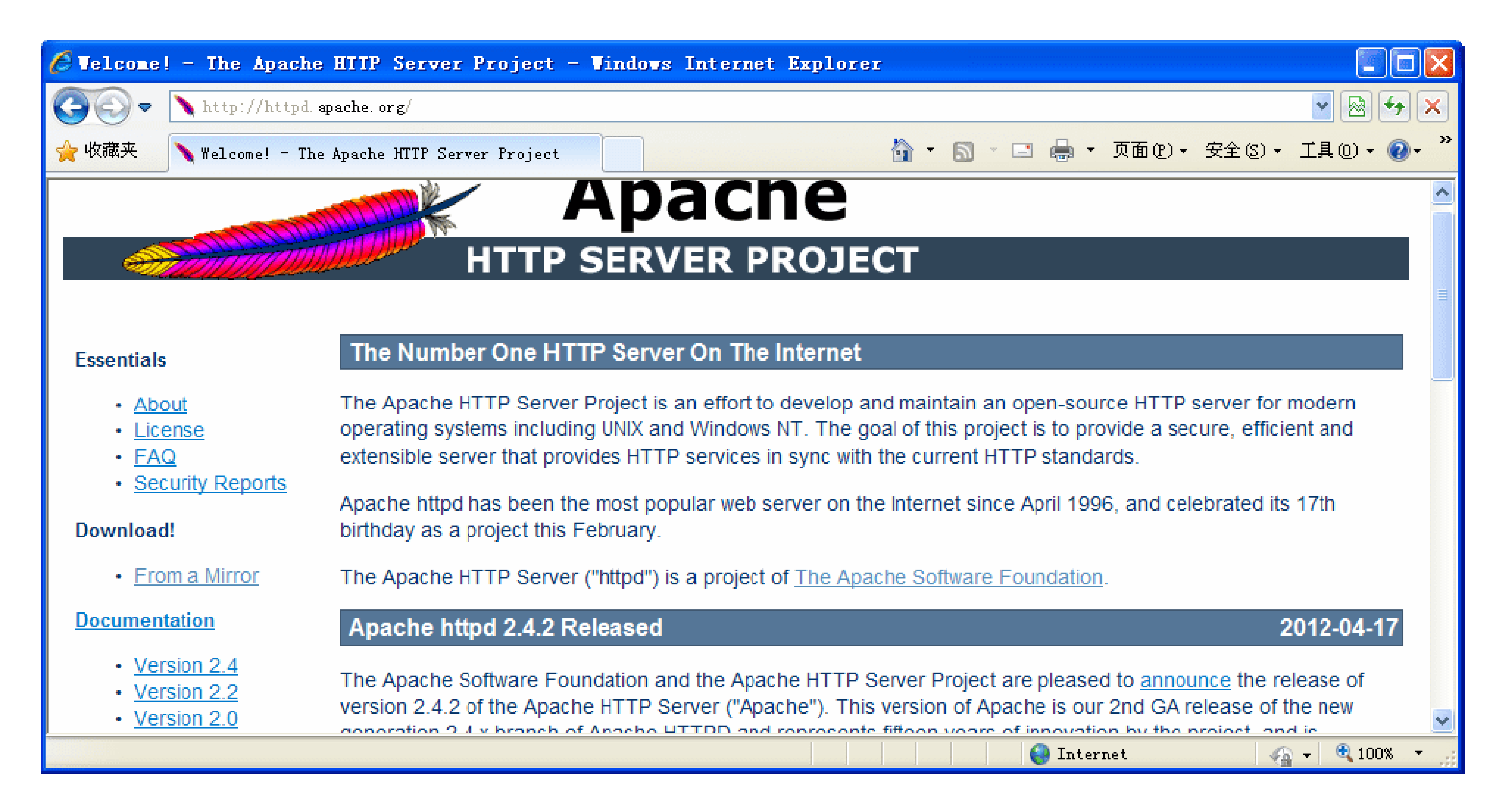


图 1-2　Apache 网站首页

【实践案例 1-1】

在 Apache 网站可以下载各个版本的 Apache、浏览修订记录或者查看帮助文档。在编写本书时，由于最新的 Apache 2.4.2 没有对应的 Windows 版本。所以以 Apache 2.2.22 版本的安装为例进行讲解。

下载后得到文件 httpd-2.2.22-win32-x86-openssl-0.9.8t.msi。双击该文件启动 Apache 安装程序向导，在安装的过程中会出现如图 1-3 所示的对话框。

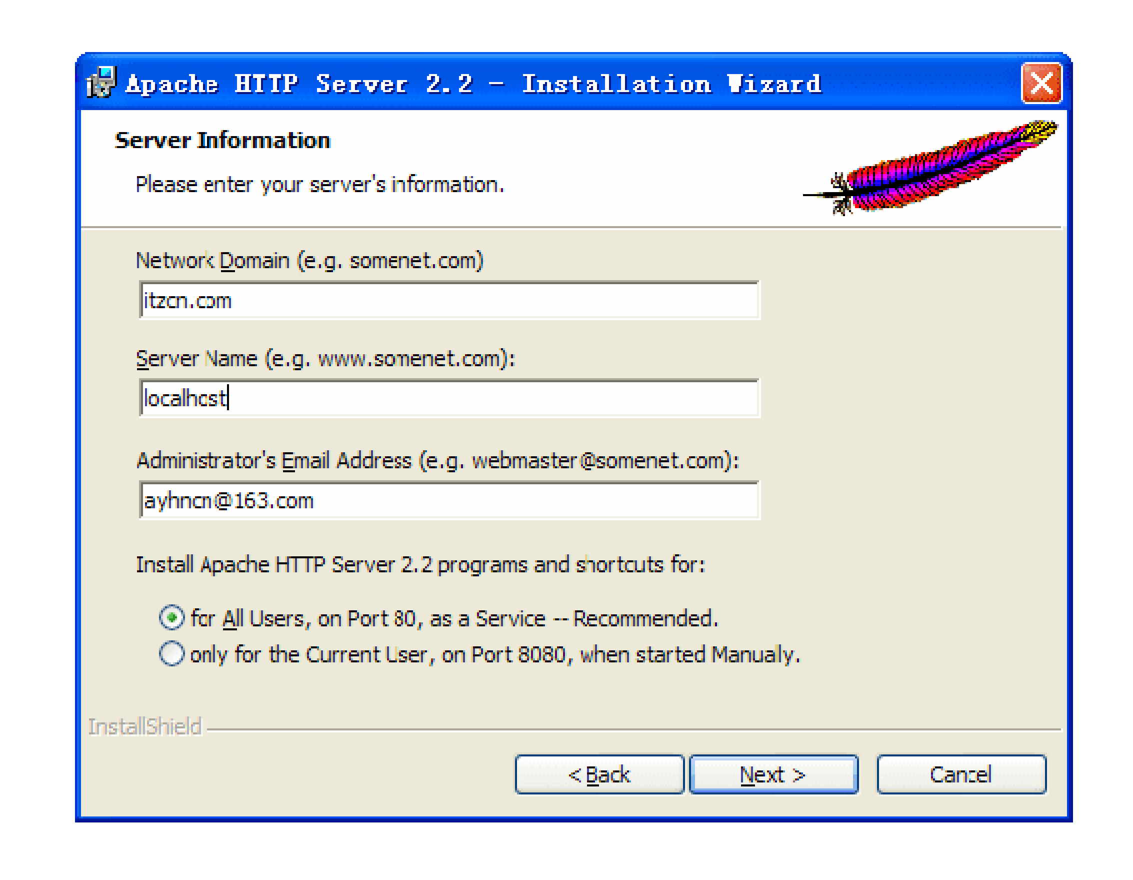


图 1-3　Aapche 选项输入

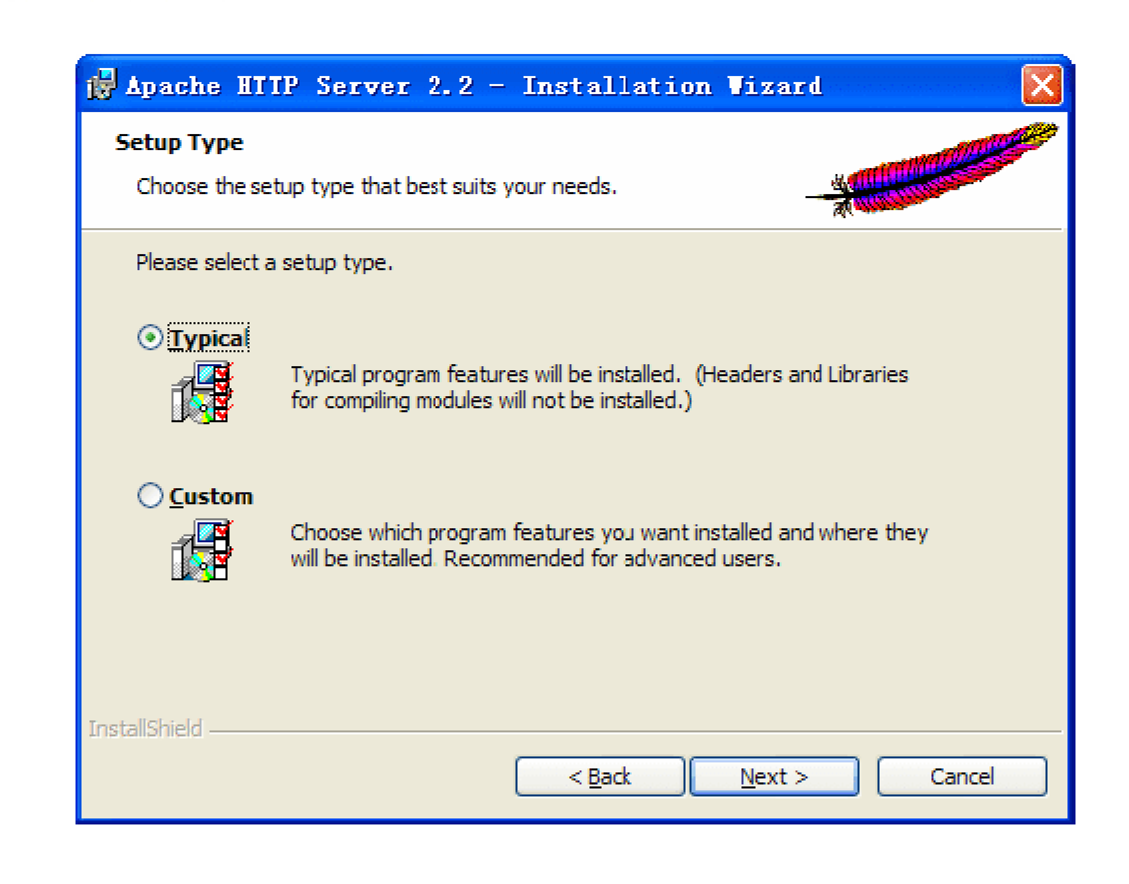


图 1-4　选择安装类型

在图 1-3 中，第一个文本框要求输入计算机的网络域，默认不填写表示是本机 IP 地址；第二个文本框表示服务器的名称，这里可以设定为 localhost；第三个文本框是系统管理员的电子邮件地址。通过单选按钮选择的是为所有用户提供 Apache 服务，还是仅为当前用户提供服务，这里使用默认配置。

单击 Next 按钮，在如图 1-4 所示的对话框中选择典型安装 Typical 选项。单击 Next 按钮，在如图 1-5 所示的对话框中，单击 Change 按钮可以设定 Apache 的安装目录。然后根据提示单击 Next 按钮，开始安装 Apache 服务器。安装完成后，将显示如图 1-6 所示的对话框。

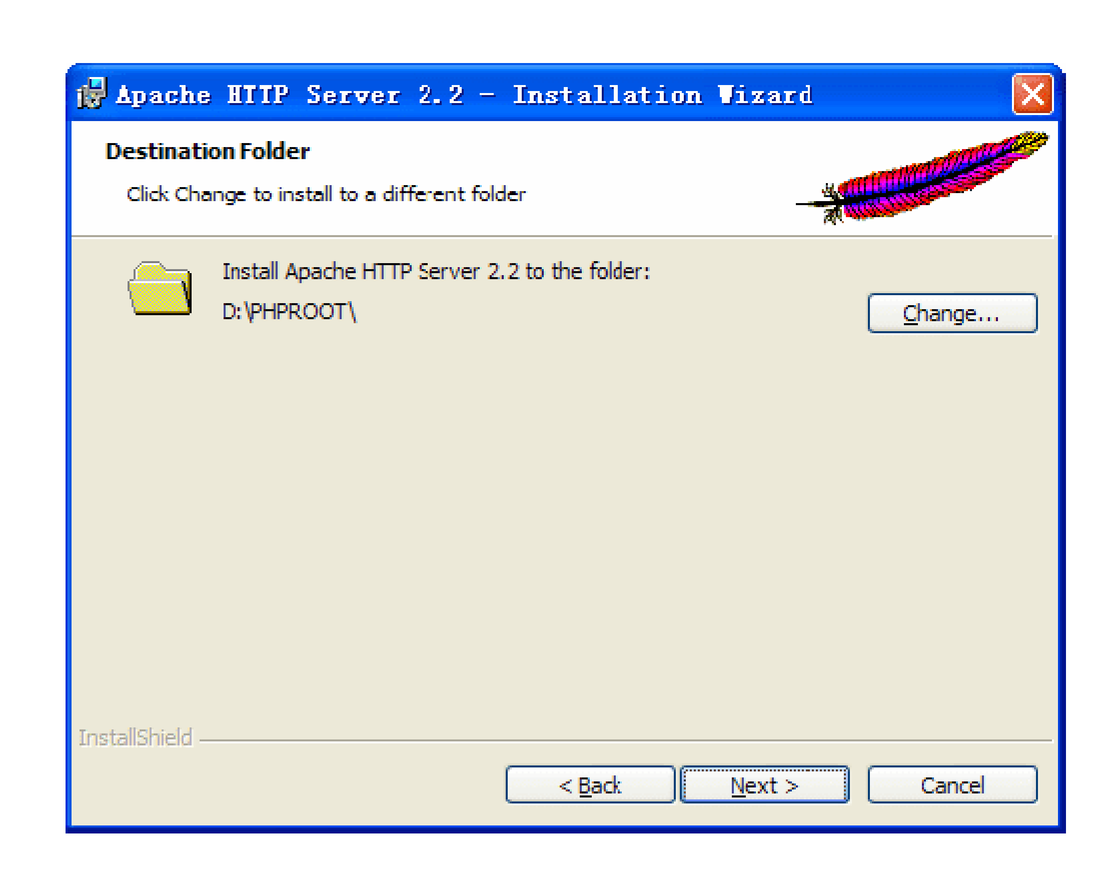


图 1-5　选择安装路径

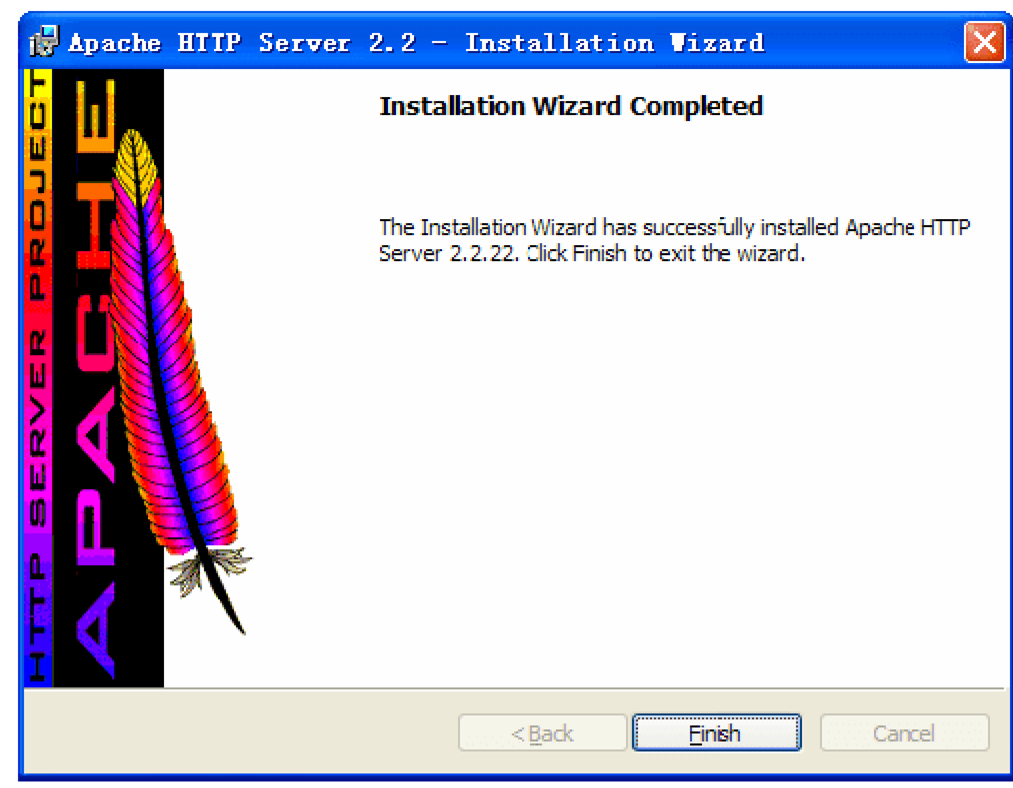


图 1-6　Apache 安装完成

这时会在右下角的状态栏出现图标，表示 Apache 已经成功安装了。这时单击该图标，会出现图标，单击该图标会出现一个级联菜单，在菜单中选择 Start 菜单项来启动 Apache。成功启动后，会出现图标。

打开 IE 浏览器，在地址栏中输入 http://localhost，如果显示如图 1-7 所示的页面说明 Apache 正常启动。

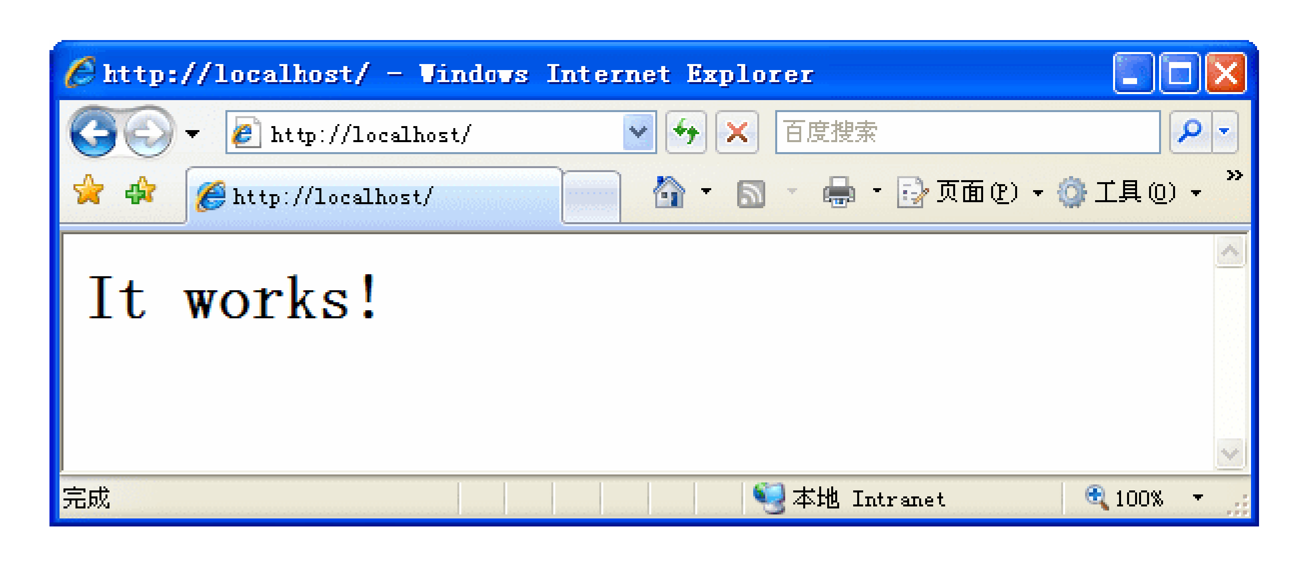
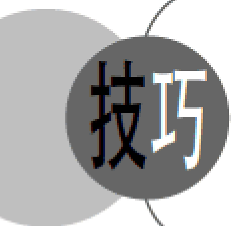


图 1-7　Apache 测试页面



还可以使用另外一种方式启动 Apache 服务器：在命令提示符窗口进入 Apache 安装后的 bin 目录下，再运行 httpd 命令。

1.2.2　安装 PHP

安装好 Apache 之后，下面开始安装 PHP。PHP 是开发和运行程序的核心，可以从 PHP 官方网站下载，地址为http://www.php.net。

在 Windows 中 PHP 有两种安装方式，如果下载的是.msi 文件则需要全新安装，如果是.zip 文件则需要解压安装。下面分别介绍这两种方式。

【实践案例 1-2】

如果读者在 PHP 官方网站下载的 PHP 非安装程序而是压缩包，那么在配置时需要设置环境变量。具体设置环境变量的方法如下。

（1）首先将 PHP 包解压到指定文件夹中作为 PHP 的根目录，例如 D:\PHPROOT\php

目录。



在选择解压的路径时，要注意路径中不能使用带有空格的路径，例如“F:\Program Files\PHP”。

（2）然后再配置 Apache 运行时需要加载的 php5apache2_2.dll 文件。方法是将 PHP 的安装路径追加到 Windows 系统中 Path 路径的下面。右击【我的电脑】图标，选择【属性】命令，在弹出的【系统属性】对话框中切换到【高级】选项卡，再单击【环境变量】按钮打开【环境变量】对话框。从【系统变量】列表中找到 Path 路径，单击【编辑】按钮后在【编辑系统变量】对话框中将 D:\PHPROOT\php 追加到路径中，如图 1-8 所示。



图 1-8　修改 Path 路径

（3）接下来进入 Apache 的 conf 目录，打开 httpd.conf 文件，在文件的最下方增加下列三行内容：

```
LoadModule php5 module "d:\phproot\php5apache2 2.dll"
AddType application/x-httpd-php .php
PHPIniDir "d:\phproot"
```

在该段代码中，第一行表示要加载的模块在哪个位置存储。第二行表示将一个 MIME 类型绑定到某个或某些扩展名。.php 只是一种扩展名，这里可以设定为.ptml、.php5 等。第三行表示 PHP 所在的初始化路径。此时 PHP 环境就配置完成了。

（4）添加完成之后，还需要重新启动 Apache 服务器查看是否成功加载 PHP 模块。如图 1-9 所示为成功后在 Apache 服务管理工具中看到的效果，在底部显示了 Apache 和 PHP 的版本。

【实践案例 1-3】

这里以 PHP 5.2.8 版本为例介绍安装过程。双击下载的 php-5.2.8-win32-installer.msi 文

件，弹出 PHP 安装程序向导，单击 Next 按钮在出现 Destination Folder 提示时单击 Browse 按钮来更改 PHP 的安装路径，这里为 D:\web\PHP\目录，如图 1-10 所示。

单击 Next 按钮选择安装 Apache 版本号，这里为 Apache 2.2x Module。再单击 Next 按钮选择 Apache 服务器的安装路径，如图 1-11 所示。然后单击 Next 按钮选择要安装的 PHP 组件，再单击 Install 按钮开始安装 PHP。安装完成后会显示完成对话框，提示成功安装了 PHP 软件包，单击 Finish 按钮完成安装。

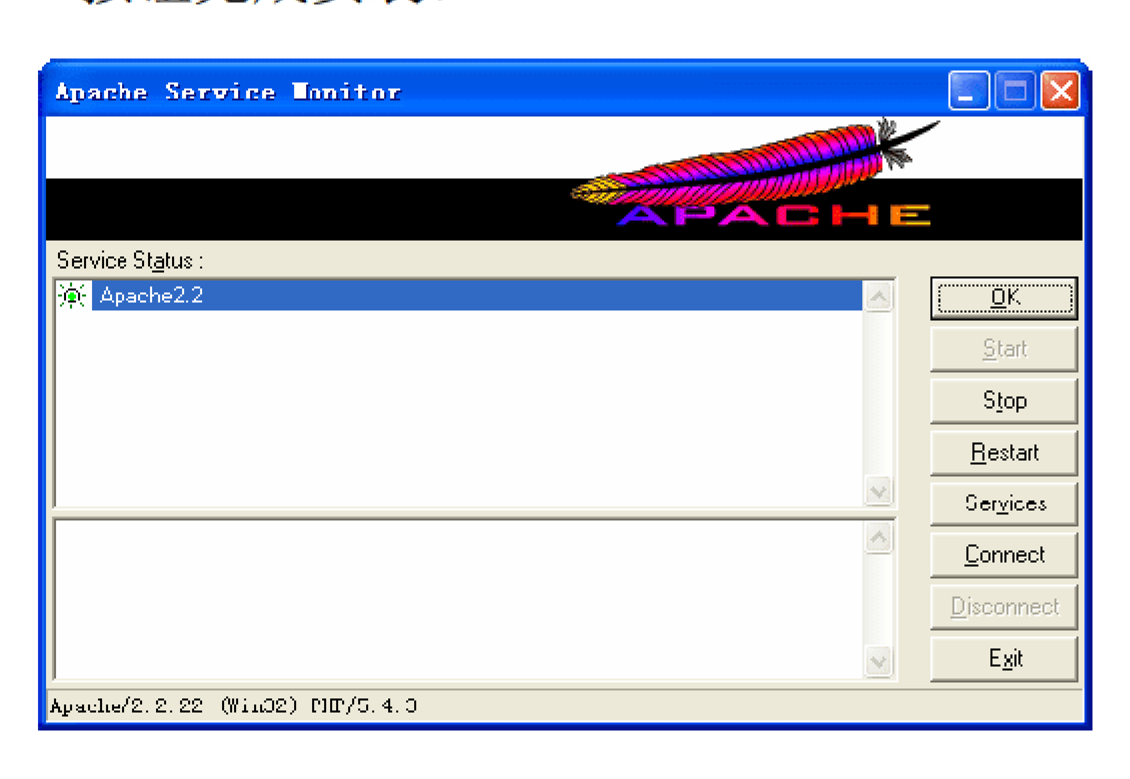


图 1-9　添加 PHP 模块后的 Apache

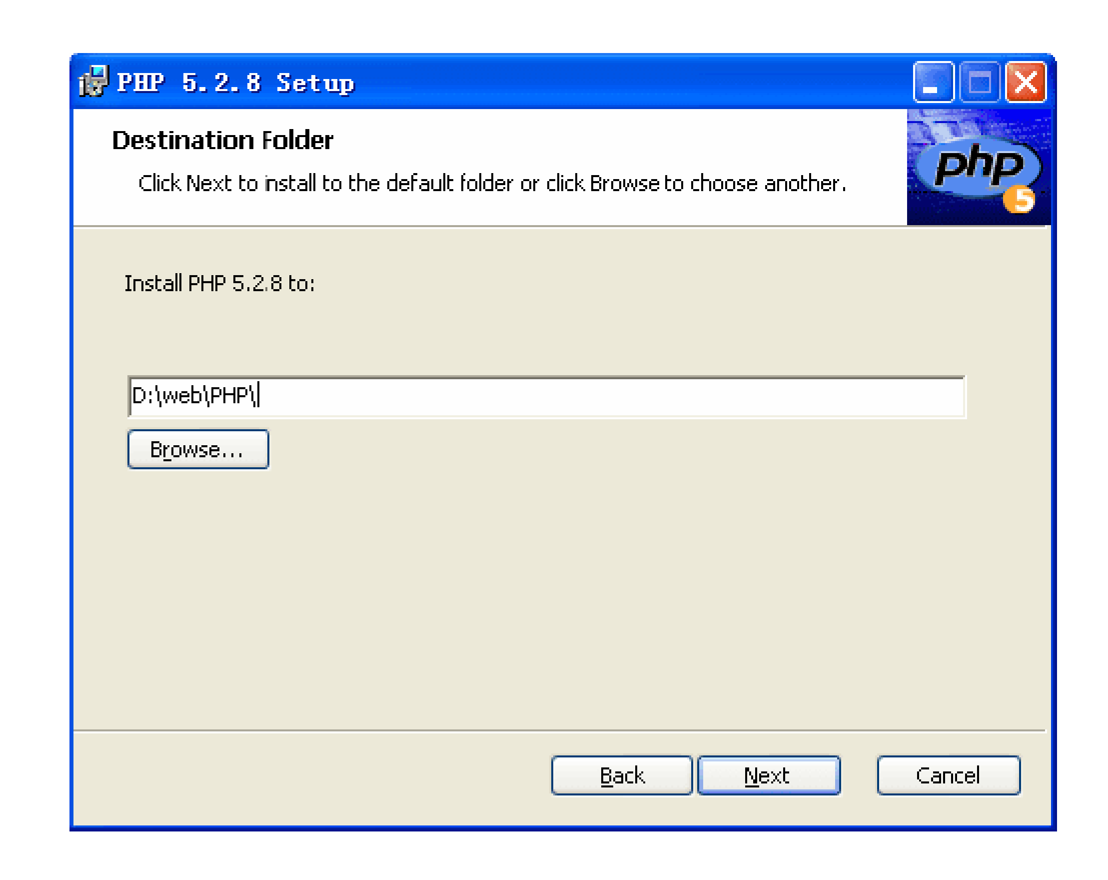


图 1-10　选择 PHP 安装路径

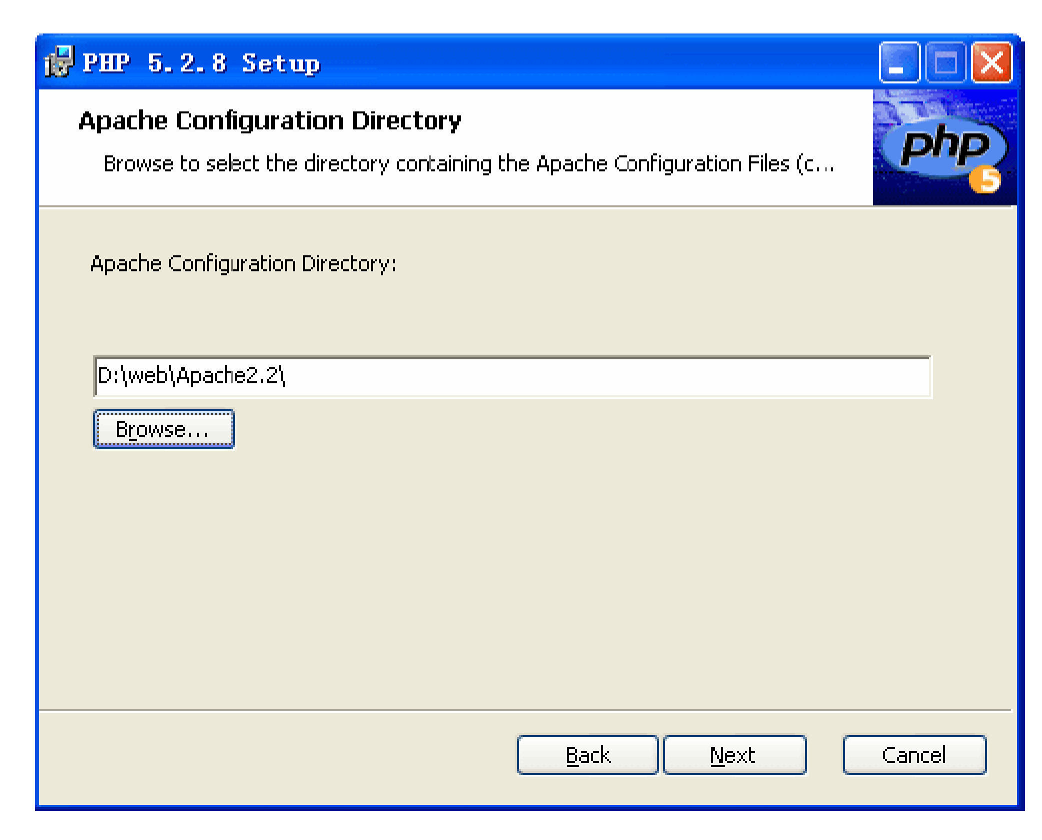


图 1-11　选择 Apache 服务器安装路径

【实践案例 1-4】

至此，运行和开发 PHP 所需的各个环境就安装完成了。下面创建一个 PHP 示例来测试 PHP 环境是否正确。示例将执行一个带有 PHP 脚本的程序，如果执行成功则证明 PHP 安装成功。

首先打开 Apache 下的 htdocs 目录（这里为 D:\PHPROOT\htdocs），然后使用记事本创建一个名为 hello.php 的文件，再添加如下代码到文件中。

```
<title>PHP 环境测试</title>
<?php
phpinfo();                                    //输出 PHP 环境信息
?>
```

保存 test.php 文件，然后在 IE 浏览器的地址栏中输入 http://localhost/hello.php，如果能够显示 PHP 的相关信息，则证明 PHP 环境配置成功，如图 1-12 所示；否则安装失败。

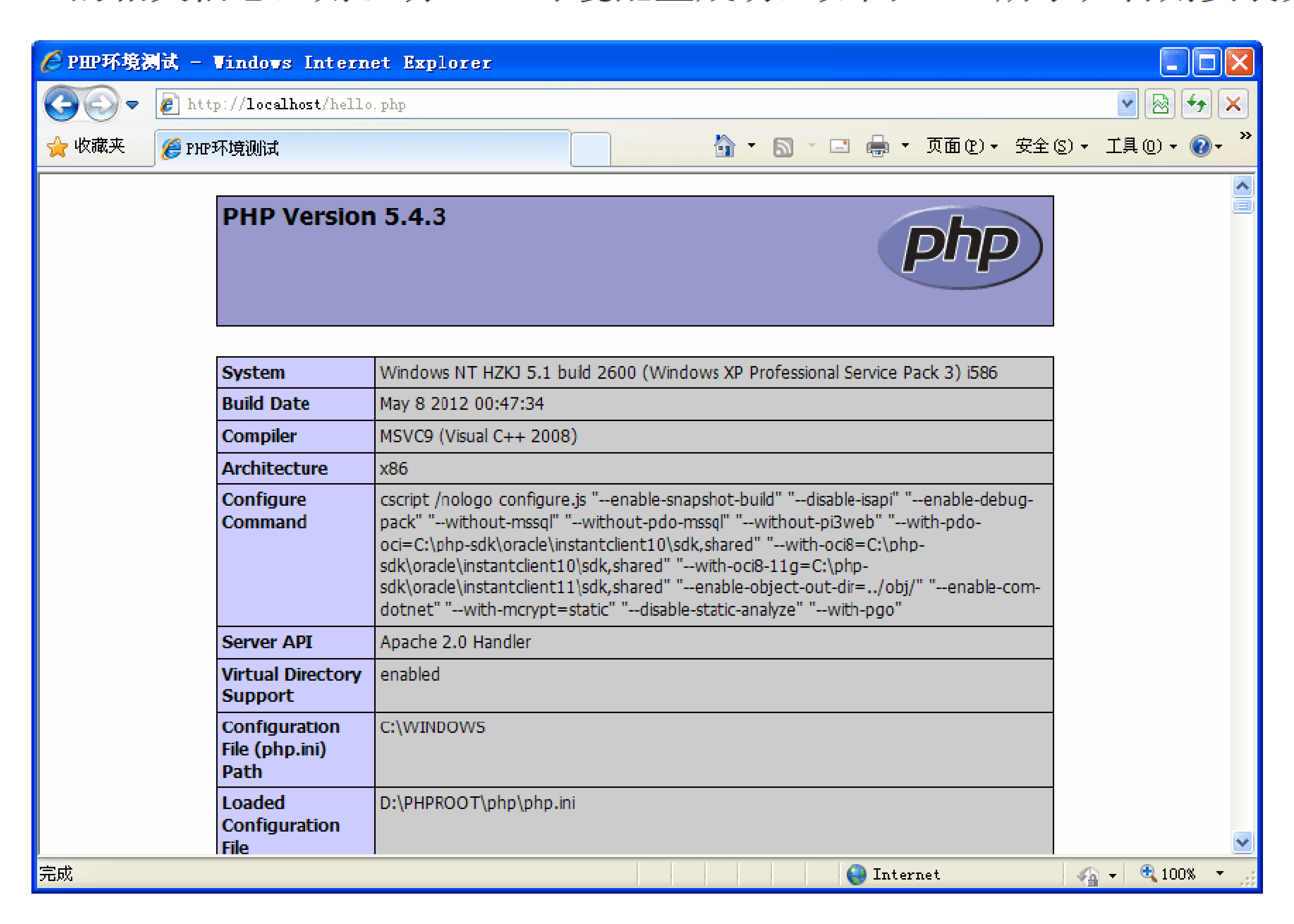


图 1-12　hello.php 页面效果

1.3　集成方式搭建 PHP 环境

与安装方式搭建 PHP 环境相比，使用集成方式的最大优点是一步到位，省去烦琐的下载、安装和配置步骤。只需要下载集成了 PHP 环境的软件包即可拥有开发 PHP 所需的一切环境，像 Web 服务器、数据库、解释器和扩展库等。

目前比较流行的 PHP 软件包有 WampServer 和 PHPnow，下面详细介绍它们的安装。

1.3.1　WampServer

WampServer 的全称是 Windows Apache MySQL PHP Server，是目前为止最完整的 Apache 服务器+PHP CGI 解释器+MySQL 数据库的整合软件包。

WampServer 的官方网站是 www.wampserver.com，在这里可以下载最新的版本。本文这里下载的是 WampServer 2.2e，其中包括 Apache 2.2.22、MySQL 5.5.24、PHP 5.3.13、XDebug 2.1.2、XDC 1.5、PhpMyadmin 3.4.10.1、SQLBuddy 1.3.3 和 webGrind 1.0。

【实践案例 1-5】

下面以 WampServer 2.2e 为例介绍安装过程。

（1）双击下载的 exe 安装文件，打开安装程序的欢迎界面，如图 1-13 所示。

（2）在欢迎界面中显示了软件包中集成的各个软件及其版本，单击 Next 按钮继续。在

进入的页面中选择 I accept the agreement 选项同意安装协议。

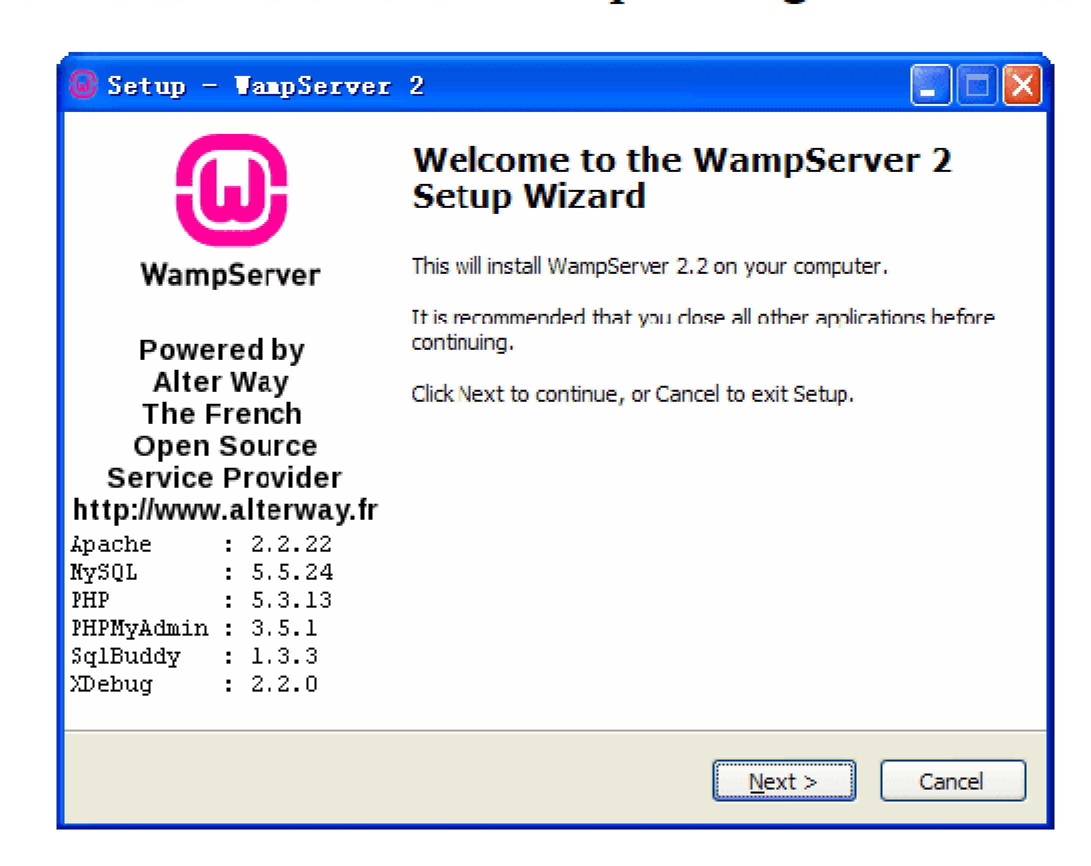


图 1-13　欢迎界面

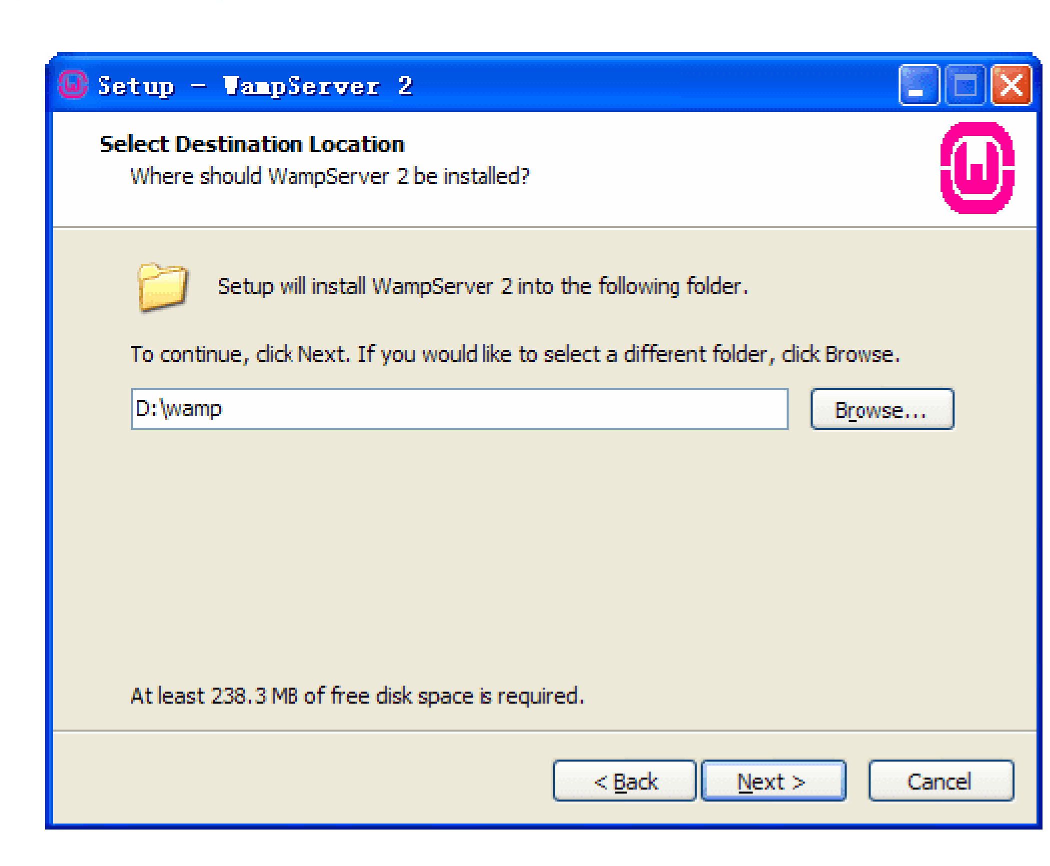


图 1-14　指定安装位置

（3）单击 Next 按钮在进入的页面中为软件包指定安装位置，默认为 C:\wamp，如图 1-14 所示。

（4）单击 Next 按钮在进入的页面中设置是否创建桌面快捷图标和快速启动图标。单击 Next 按钮进入确认安装页面，如图 1-15 所示。

（5）确认无误后单击 Install 按钮开始安装。在安装结束后会提示用户选择使用的默认浏览器，如果不需要直接单击【打开】按钮即可，如图 1-16 所示。

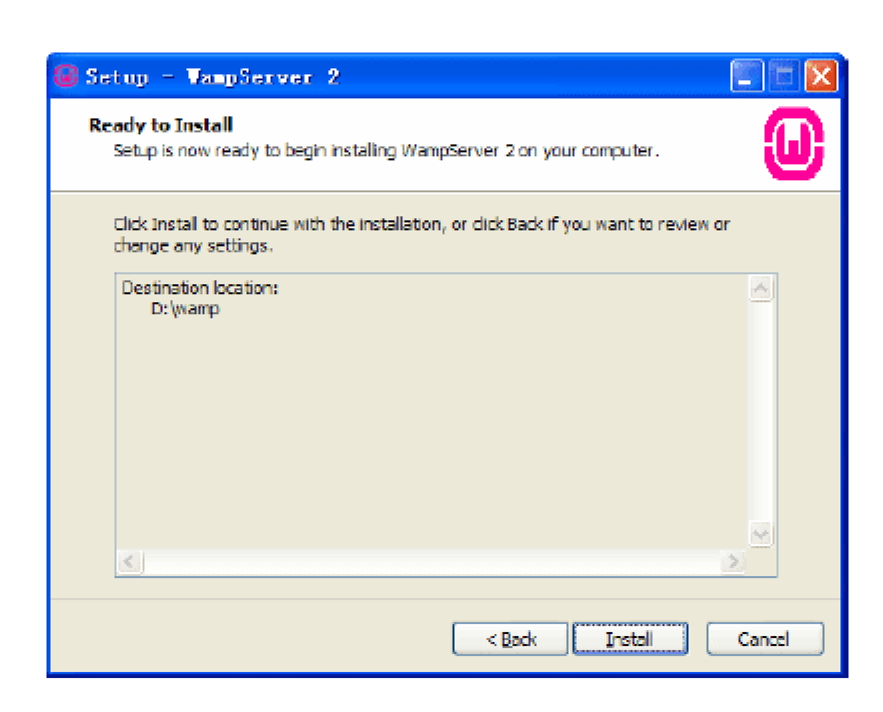


图 1-15　确认安装

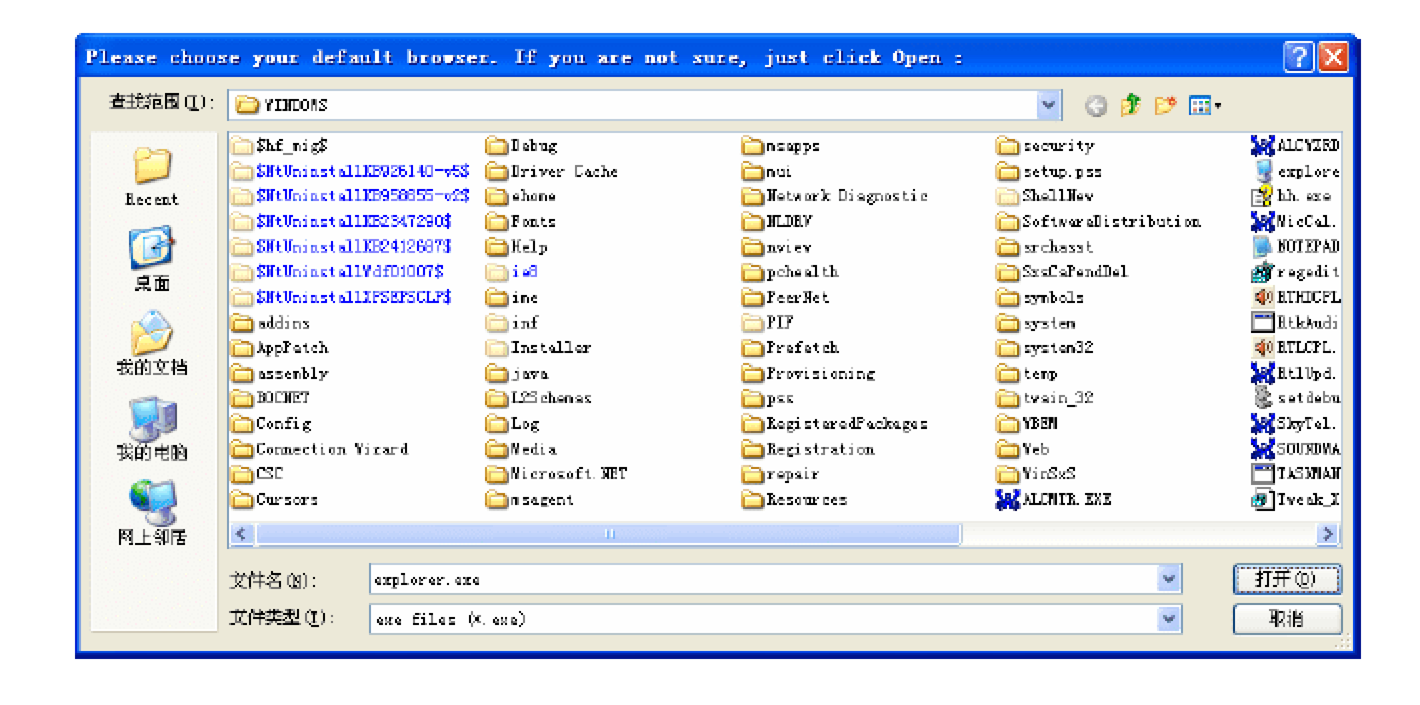


图 1-16　选择浏览器位置

（6）安装完成之后还需要设置 PHP 的邮件参数，一般使用默认值即可，如图 1-17 所示。

（7）单击 Next 按钮，如图 1-18 所示，在最后一步中单击 Finish 按钮结束安装。

经过上面步骤完成 WampServer 的安装之后，会在系统状态栏中看到图标。默认菜单使用英文显示，可以右击该图标选择 Language | Chinese 菜单项更改为中文显示。



如果装有同类软件，应先停止或卸载，否则会占用端口！需关闭迅雷，或修改迅雷的 BT 端口！

WampServer 安装完成之后，默认的 Web 根目录位于安装位置的 www 目录下。为了测试 WampServer 是否工作正常，可以在浏览器中输入 http://localhost 进行验证。如果看到如

图 1-19 所示的效果说明运行正确。

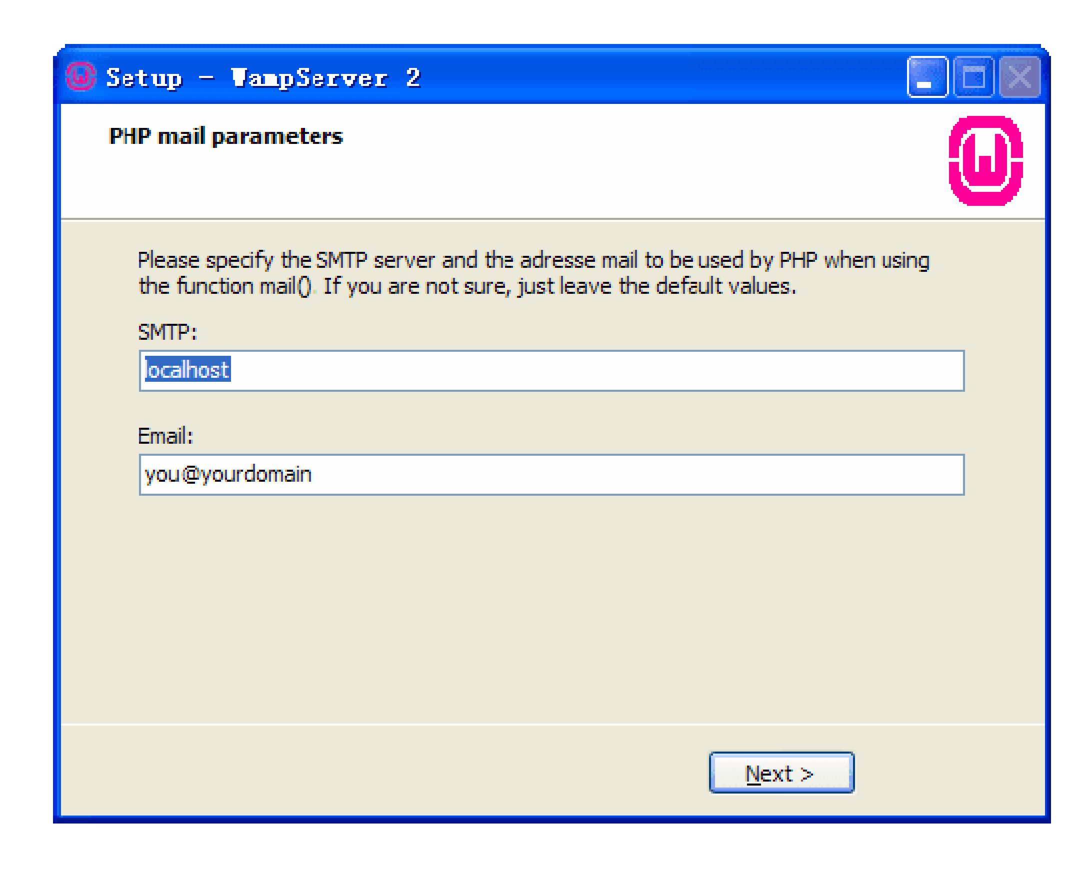


图 1-17　设置邮件参数

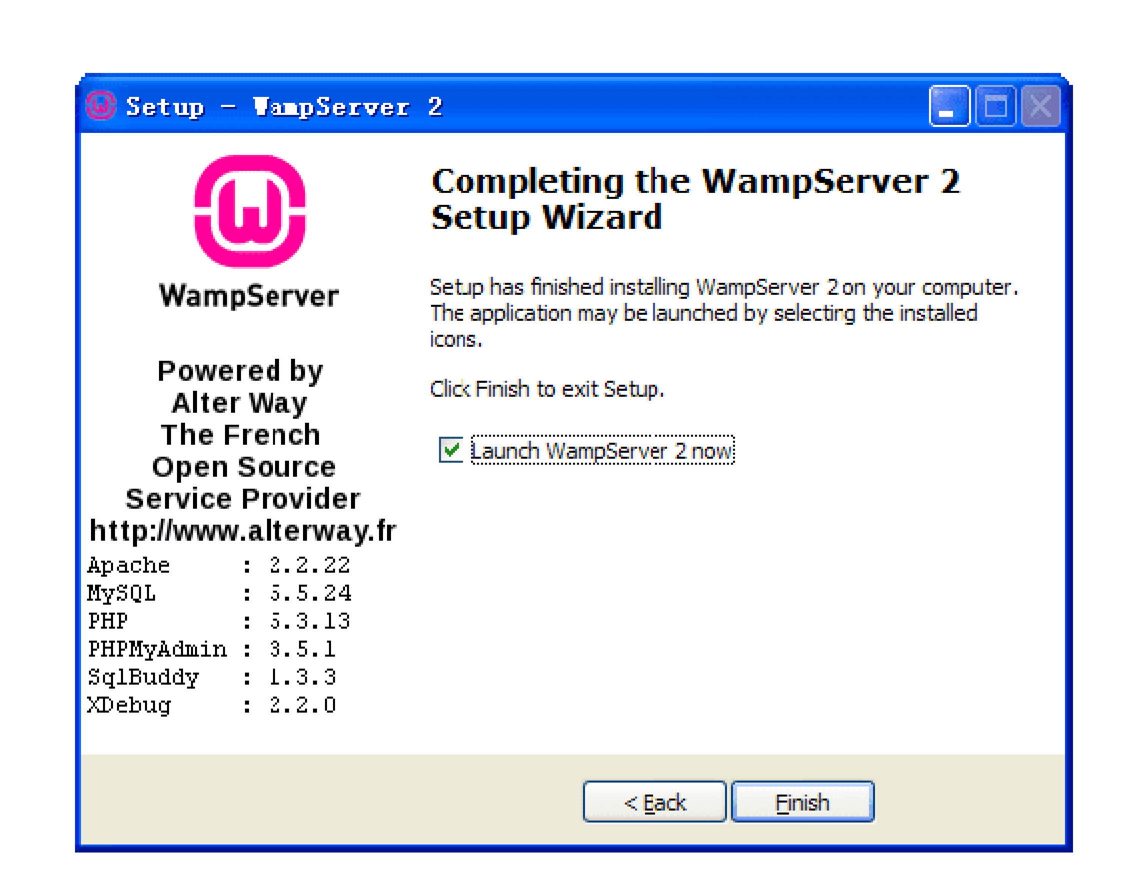


图 1-18　结束安装

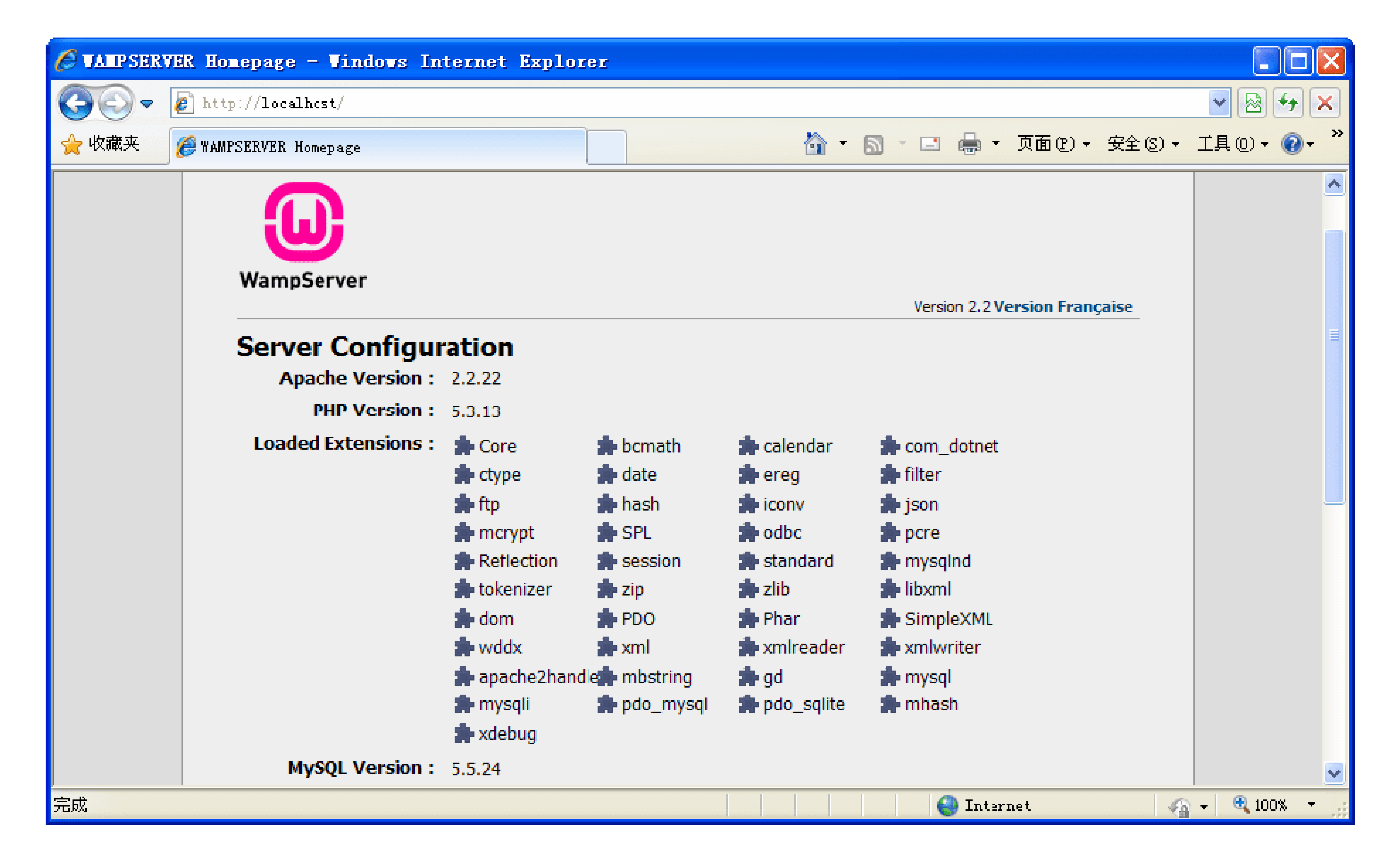
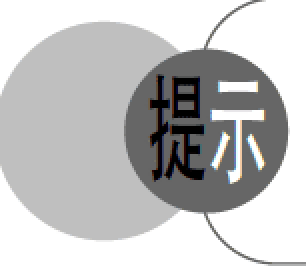


图 1-19　测试 WampServer



WampServer 还有很多参数设置，由于篇幅有限这里就不再详解。有兴趣的读者可以自己试一试。

1.3.2　PHPnow

搭建 PHP 其实不难，只是有点烦琐。自己搭建一次 PHP+MySQL 环境很费时。更糟的是，很多新手在配置 PHP 时常常出现问题，像 mysql 扩展、zend 安装失败等问题。这时，我们需要一个快速、标准且专业的 PHP 软件包。PHPnow 就这样应运而生，为用户快速搭建专业的 PHP 环境。

PHPnow 是绿色免费的 PHP 软件包，使用它可以简易安装、快速搭建支持虚拟主机的 PHP 环境。附带 PnCp.cmd 控制面板可以帮助用户快速配置各种参数，使用非常方便。

PHPnow 的官方网站是 http: //phpnow.org，目前最新版本是 PHPnow 1.5.6，其中包括 Apache 2.0.63、PHP 5.2.14、MySQL 5.0.90、Zend Optimizer 3.3.3 和 phpMyAdmin 3.3.7。

【实践案例 1-6】

PHPnow 是绿色的，解压后执行 Setup.cmd 初始化，即可得到一个 PHP+MySQL 环境。然后就可以直接安装 Discuz、PHPWind 或者 WordPress 等 PHP 程序。

下面以 PHPnow 1.5.6 为例介绍安装过程。

（1）从 http://phpnow.org 网站下载 PHPnow 1.5.6，得到一个名为 PHPnow-1.5.6.zip 的文件。

（2）解压后执行 Setup.cmd 文件，首先提示选择 Apache 的版本，如图 1-20 所示。

```
PHPnow - 组件版本选择

选择 Apache 版本
 20      Apache 2.0.63 <推荐>
 22   -  Apache 2.2.16
-> 请选择: 22
```

图 1-20　选择 Apache 版本

（3）然后提示选择 MySQL 版本，之后解压文件到指定目录。接着程序提示将会调用 Init.cmd 初始化，如图 1-21 所示。

```
PHPnow - 正在解压 ...
Extracting  Apache-22\conf\extra
Extracting  Apache-22\conf
Extracting  Apache-22\cgi-bin
Extracting  Apache-22\bin\iconv
Extracting  Apache-22\bin
Extracting  Apache-22

Everything is Ok

Folders: 124
Files: 2477
Size:       161455526
Compressed: 18827256

 # 解压完成! 是否执行 Init.cmd 初始化?
-> 请选择(y/n): y
```

图 1-21　初始化

（4）初始化完成之后会让用户输入 MySQL 的管理密码。完成安装将看到如图 1-22 所示界面。



图 1-22　完成安装

（5）在浏览器中输入 http://127.0.0.1 测试 PHPnow 是否正常，如图 1-23 所示。

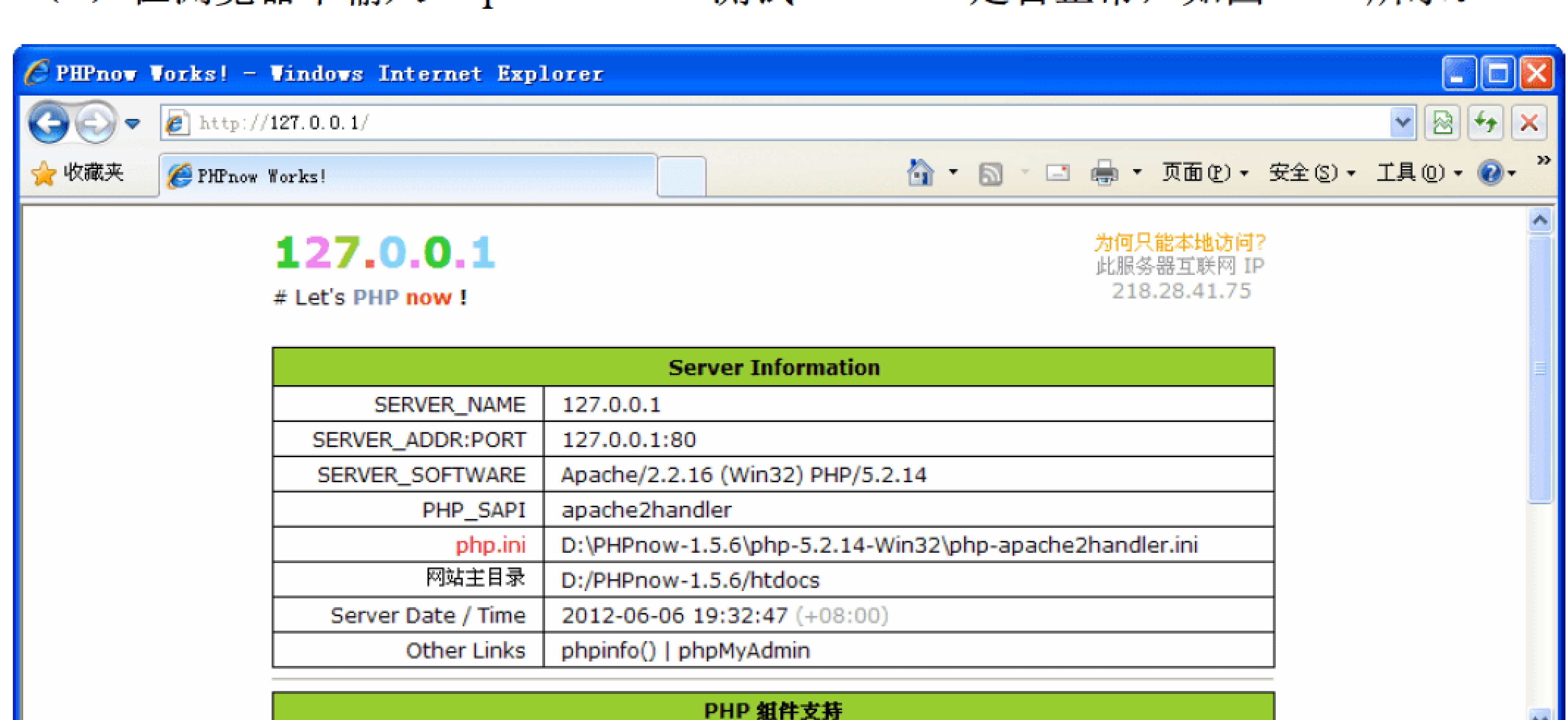
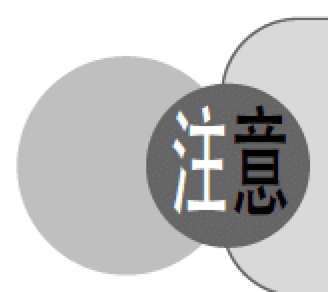


图 1-23　测试 PHPnow

PnCp.cmd 是 PHPnow 的控制面板，大部分管理功能都在上面实现。在 PnCmds 目录下有一些常用的 cmd 脚本用于控制程序的运行。在使用时可以为它们建立快捷方式到桌面或其他位置。



成功初始化后 Init.cmd 自动重命名为 Init.cm_。如有必要，可将其改名为 Init.cmd 再重新初始化。重新初始化不会丢失网站数据，仅仅是恢复最初的配置信息。

1.4　查看 PHP 配置文件

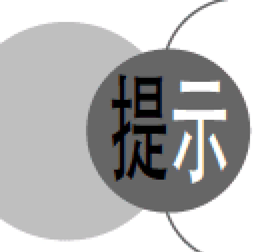
PHP 的 Windows 安装总共有 45 个扩展包，都位于 PHP 安装位置的 ext 目录下。通过这些包的使用，可以增加一些新的功能。但是，要真正使用这些扩展包，还需要对 PHP 的配置文件进行修改。

PHP 的配置文件为 php.ini，保存在 PHP 的安装目录下。php.ini 遵循许多 Windows 应用程序中 INI 文件的常见结构，是一个 ASCII 文本文件，并且被分成几个不同名称的部分，每一部分包括与之相关的各种变量。每一部分类似于如下结构：

```
[SectionName]
variable="value"
anothervariable="anothervalue"
```

各部分的名称通过方括号"[]"括起来放在顶部，然后将是任意数量的“变量名=值”对，每一对占单独一行。如果行以分号";"开头则表明该行是注释语句。

如下所示为 php.ini 文件中的片段内容：



在 php.ini 中启用或禁止 PHP 功能非常简单。只需要将相关语句注释而无须删除，该语句就不会被系统解析。特别是当希望在一段时间以后重新打开某种功能的时候特别方便，因为不需要在配置文件中将此行删除。

```
 [PHP]
; Enable the PHP scripting language engine under Apache.
engine = On
; Allow the <? tag.  Otherwise, only <?php and <script> tags are recognized.
; NOTE: Using short tags should be avoided when developing applications or
; libraries that are meant for redistribution, or deployment on PHP
; servers which are not under your control, because short tags may not
; be supported on the target server. For portable, redistributable code,
; be sure not to use short tags.
short_open_tag = On
; Allow ASP-style <% %> tags.
asp tags = Off
; The number of significant digits displayed in floating point numbers.
precision    =  14
```

在这个文件中可对 PHP 的 12 个方面进行设置，包括：语言选项、安全模式、语法突出显示、杂项、资源限制、错误处理和日志、数据处理、路径和目录、文件上传、Fopen 包装器、动态扩展和模块设置等。由于篇幅有限，在这里就不再详细介绍每个方面的选项设置。



由于每次启动 PHP 时都会读取 php.ini。因此，在修改 php.ini 文件改变 PHP 配置之后，需要重启 Web 服务器以使配置的改变生效。

1.5　选择 PHP 语法风格

经过前面的学习，读者应该已经具备开发和运行 PHP 的环境。因为 PHP 是嵌入 HTML 的脚本语言，因此，为了区别 HTML 的标记和 PHP 脚本，还需要把嵌入的 PHP 脚本语言放置到特定的、成对的标记内。这样当解析一个 PHP 文件时，会寻找相应的开始和结束标记，这些标记告诉 PHP 解析器开始和停止的位置。在这对开始和结束标记之外的内容会被 PHP 解析器忽略。

PHP 提供了 4 种在 HTML 页面嵌入 PHP 脚本的方式，下面分别介绍。

1.5.1　默认标记

PHP 的默认标记使用“<?php”作为开始，使用“?>”作为结束，例如如下代码所示：

```
<?php
    echo "PHP 的默认标记，使用方法非常简单";
?>
```

代码在运行时，PHP 解析器只解析开始标记“<?php”和结束标记“?>”中的代码，即使用 echo 输出函数输出“PHP 的默认标记，使用方法非常简单”。

1.5.2 ASP 风格标记

ASP 风格标记以“<%”开始，以“%>”结束，对于了解 ASP 的读者应该很熟悉这种风格的标记。PHP 同样也支持这种编写方式，例如如下代码所示：

```
<%
    echo "PHP 支持使用 ASP 风格的标记";
%>
<%= "一段 PHP 脚本"%>
```



要使用 ASP 风格标记需要修改配置文件。它的配置选项是 asp_tags，设置为 on 表示启用。

1.5.3 脚本标记

脚本标记方式是指使用类似于 JavaScript 的语法嵌入 PHP 脚本，即以“<script language="php">”开始，以“</script>”标记结束，例如如下代码所示：

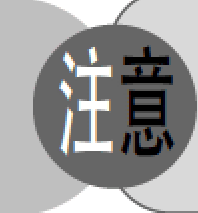
```
<script language="php">
    echo "<h3>欢迎使用 PHP<h3>";
    echo "PHP 的语法非常灵活，这是采用 PHP 脚本标记输出的。";
</script>
```

1.5.4 短标记

PHP 还支持一种使用更短的方式嵌入 PHP 脚本，因此被称为短标记。

短标记以“<?”开始，以“?>”结束，其中省略了默认标记中必需的 php 引用，例如如下代码所示：

```
<?="这些内容是使用短标记输出的"?>
<?
   echo "这些内容是使用短标记输出的";
?>
```



要使用短标记方式，必须在 PHP 配置文件中将 short_open_tag 选项的值设置为 on。默认为 off，表示禁用短标记。

【实践案例 1-7】

上面详细讲解了每种方式的语法及其使用示例。当编写 PHP 脚本时，可以在一个页面混合使用它们中的一种或者几种。例如，下面将创建一个 PHP 页面，同时使用这 4 种方式输出一个 HTML 页面。

（1）在 Apache 的 htdocs 目录下创建一个名为 tags.php 的文件。

（2）用记事本打开 tags.php，然后将下面所示的代码添加到文件中。

```
<?php
$siteName="窗内网";
?>
<html>
<head>
<title><%=$siteName%>简介</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<h3>词条名称: <%=$siteName%></h3>
<script language="php">
  echo "<img src='images/logo.gif'>";
  echo "<br/><br/>";
</script>
简介:
<%
  echo $siteName."是由汇智计算机技术服务有限公司创建的一个免费视频教程网站，成立于
  2002 年。<br/>";
  echo $siteName."结合软件开发和网站建设的实际需求，投入大量资源，录制了很多精品 IT
  教学视频教程，包括 SQL Server、PHP、Java、Eclipse、Struts、C#、Oracle 等开发技
  术，以及 3ds max、Maya 等三维动画与视频处理教学视频产品等。<br/>";
  echo "公司与清华大学出版社合作编写了《完全学习手册》、《网络大讲堂》、《清华学堂》等系列
  图书，得到了读者的一致好评和认可！<br/>";
  echo $siteName."官方网站 http://www.itzcn.com";
%>
<?php
echo "<hr/>";
echo "提示:本页面主要用于演示 PHP 中 4 种语法标记的使用,需要在 PHP 配置文件中将 asp_tags
和 short_open_tag 选项的值设置为 on。";
?>
</body>
</html>
```

（3）为了使代码中的各种嵌入方式都有效，需要打开 PHP 的配置文件 php.ini，将 asp_tags 和 short_open_tag 选项的值都设置为 On。

（4）保存对 php.ini 的修改，然后重新启动 Apache。

（5）打开浏览器，输入地址 http://localhost/tags.php 查看运行效果，如图 1-30 所示。

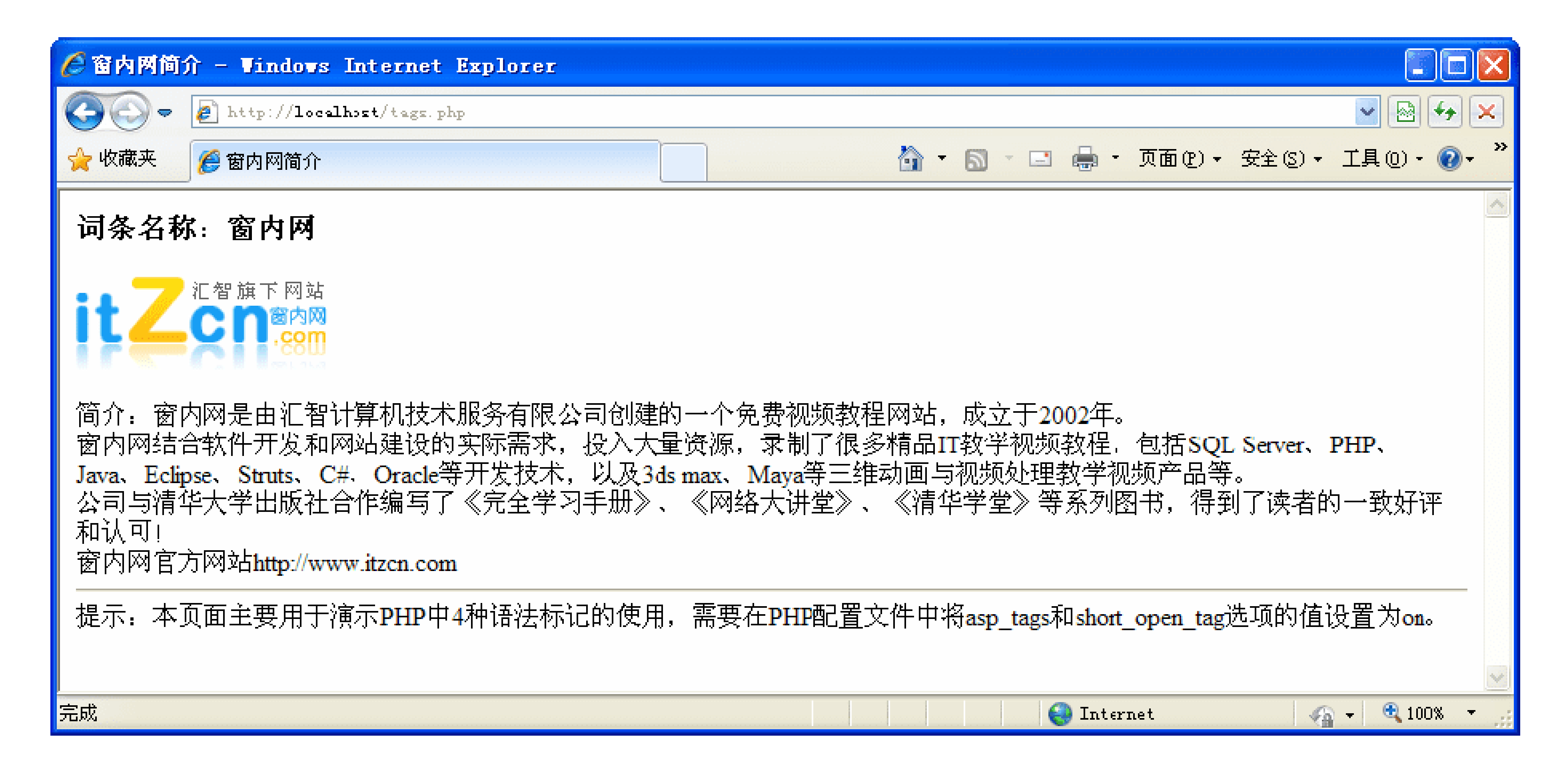


图 1-30　tags.php 运行效果

1.6　向页面输出内容

PHP 脚本运行之后最终需要显示到页面，这就需要用到输出函数。在前面多次使用的 echo 就是 PHP 的输出函数之一。

PHP 输出函数的功能非常实用也很强大，它们都用于把 PHP 处理过的信息显示到客户端。本节将介绍 PHP 最常用的两类输出函数。

1.6.1　输出字符串

这类函数会将字符串的内容原样输出，不进行任何的转义和格式化。最常用的就是 print()函数和 echo()函数。

1. print()函数

print()函数用于向页面输出信息，为用户提供信息反馈，支持输出字符串和变量。它的返回值是一个布尔类型，如果输出成功返回 true，否则返回 false。

语法格式如下所示：

```
boolean print(argument)
```

【实践案例 1-8】

根据上面的语法格式创建一个示例来讲解 print()函数的具体使用，代码如下：

```
<?php
    print("大家好。\n 我是一个初学者，非常喜欢 PHP。\n ");
    $siteName="www.itzcn.com ";
    print($siteName."是一个不错的学习网站，推荐你去看看。");
?>
```

在以上代码中，首先调用 print()函数输出了一个字符串。第二行创建了一个变量 $siteName，第三行使用 print()函数将变量与字符串进行连接并输出。

运行后输出结果如下：

```
大家好。
我是一个初学者，非常喜欢 PHP。
www.itzcn.com 是一个不错的学习网站，推荐你去看看。
```



在第二个 print()函数中使用的圆点“.”符号是 PHP 的字符串连接符。

2. echo()函数

echo()函数与 print()函数作用相同，都用来向客户端输出信息。但两者又有区别，首先 echo()函数不能用在复杂表达式中，因为它返回 void，而 print()返回一个 boolean 值。其次，echo()能输出多个字符串，而 print()则不能。

echo()函数的语法格式如下：

```
void echo (String $arg1 [, String $...] )
```

【实践案例 1-9】

根据上面的语法格式创建一个示例来讲解 echo()函数的具体使用，代码如下：

```
<?php
    $day="星期六";
    $time="中午 2 点";
    echo "记事本\n";
    echo $day."学校有个公开课。\n","时间: ".$time;
?>
```

运行后输出结果如下，可以看到在 echo()函数中使用多个字符串也能正确输出。

```
记事本
星期六学校有个公开课。
时间: 中午 2 点
```



echo()和 print()在功能上可以互换，但是 echo()在执行速度上稍快一些。

1.6.2 格式化输出字符串

使用格式化输出函数可以对字符串中的特殊字符进行格式化，例如将字符串中的数字转换为小数形式或者二进制等。最常用的格式化输出字符串函数有 printf()和 sprintf ()。

1. printf()函数

printf()函数用于向客户端输出一个格式化过的字符串，语法格式如下所示：

```
int printf ( string $format [, mixed $args [, mixed $...]] )
```

函数返回值是一个整型数值，表示字符串的长度。其中，$args 表示指定的参数值，它的输出将根据$format 进行格式化；$format 参数用于控制数据输出的格式，包括对齐方式、精度、类型和位置等。

$format 参数由五部分组成，它们都是可选的，按照以下先后顺序出现。

- **填充提示符　这一部分确定为达到正确的字符串大小所用的填充字符。默认为空格，也可以指定其他填充提示符（在字符前加一个单引号）。**
- **对齐提示符　这一部分确定输出是左对齐还是右对齐。默认为右对齐，可以用一个负号设置为左对齐。**
- **宽度提示符　这一部分确定此函数输出的最少字符数。**
- **精度提示符　确定应显示的小数位数，只影响浮点数类型的数据。**
- **类型提示符　这一部分确定如何转换参数，它所支持的类型提示符如表 1-1 所示。**

表 1-1　format 类型提示符

类型提示符	描述
%b	将参数认为是一个整数，显示为二进制数
%c	将参数认为是一个整数，显示为对应的 ASCII 字符
%d	将参数认为是一个整数，显示为有符号十进制数
%f	将参数认为是一个浮点数，显示为浮点数
%o	将参数认为是一个整数，显示为八进制数
%s	将参数认为是一个字符串，显示为字符串
%u	将参数认为是一个整数，显示为无符号十进制数
%x	将参数认为是一个整数，显示为小写的十六进制数
%X	将参数认为是一个整数，显示为大写的十六进制数

【实践案例 1-10】

创建一个案例演示使用 printf()函数输出不同格式的字符串，代码如下所示：

```
<?php
    $price=38.9;
    $num=11;
    printf("%.4f \n", $price);
    printf("%.1f \n", $price);
    printf("将%2d 转换为其他进制\n",$num);
    printf("%s|%d|%b|%x|%o|%f \n", $num, $num, $num, $num, $num, $num);
    printf("3.1415926*5*5=%.1f \n",3.1415926*5*5);
    printf("%s 在 %s 公司工作了 %.2f 年，约%d 个月。 \n","小桐","长城科技
    ","3.2",3.2*12);
    printf("%s 好吗，%s 很好。 \n","你","我");
    $length=printf("Hello,my name is %s,i am a %s","som","Chinese");
```

```
    printf("\n 上一个字符串的长度是: %d",$length);
?>
```

运行后输出结果如下：

```
38.9000
38.9
将 11 转换为其他进制
11|11|1011|b|13|11.000000
3.1415926*5*5=78.5
小桐 在 长城科技 公司工作了 3.20 年，约 38 个月。
你好吗，我很好。
Hello,my name is som,i am a Chinese
上一个字符串的长度是: 35
```

另外，使用 printf()函数还可以改变参数的输出顺序。例如，%2$表示位于参数列表的第 2 个参数；%3$表示位于参数列表的第 3 个参数。但是，在$format 参数的字符串中，美元$符号必须转义为\$。例如：

```
<?php
    printf("今天是%2\$s，昨天是%1\$s，明天是%3\$s", "星期一", "星期二","星期三");
?>
```

运行后输出结果如下：

```
今天是星期二，昨天是星期一，明天是星期三
```

2. sprintf()函数

sprintf()函数和 printf()函数一样，都是输出格式化后的信息，但它将结果以字符串形式返回，而不是直接按照标准输出，其语法格式如下：

```
string sprintf ( string $format [, mixed $args [, mixed $...]] )
```

例如，下面代码演示了使用 sprintf()函数的输出语句：

```
<?php
    $str = sprintf("%s 网站%d 周年店庆，所有商品全部%.1f 折。快去抢购吧！" , "窗内
    网" ,5, 0.75*100);
    echo($str);
?>
```

输出结果如下：

```
窗内网网站 5 周年店庆，所有商品全部 7.5 折。快去抢购吧！
```

1.7 程序注释

程序注释是指在程序中不能执行、用于解释说明或者描述程序功能的语句。对于初学

PHP 的读者来说，养成在代码中添加注释是一个很有必要地好习惯。一来方便日后自己调试代码，另一方面也方便他人阅读。

PHP 中的注释主要可以分为单行注释和多行注释，下面详细介绍它们。

1.7.1　单行注释

单行注释适用于内容不超过一行的情况。因为这种注释很短，因此没有必要区分这种注释的结束。

在 PHP 中单行注释与 C++中的单行注释相同，因此也可以称为单行 C++注释。即在行前添加注释符“//”。例如如下所示代码：

```
<?php
    //这是 PHP 的单行注释
    echo "这是一行 PHP 脚本。";
    echo "今天天气不错哦。";
?>
```

运行后的输出结果如下：

```
这是一行 PHP 脚本。今天天气不错哦。
```

除了注释符“//”，PHP 还可以使用符“#”来注释一行。例如，可以将上面的示例重写为如下代码：

```
<?php
    #这是 PHP 的单行注释
    echo "这是一行 PHP 脚本。";
    echo "今天天气不错哦。";
?>
```

1.7.2　多行注释

通常有一些详细的功能描述或其他解释说明需要放在 PHP 程序中，而这些说明可能包括多行。此时，便可以使用 PHP 的多行注释，它包含注释的开始和结束标记，被注释的内容以“/*”标记开始，以“*/”结束。

例如如下所示代码：

```
<?php
    /*
    多行注释
    创建时间：2012-06-19
    版本：0.1.45
    功能：数据库的实体类
    */
```

```
    echo "实体类";
?>
```

在运行时，PHP 解析器将直接跳过“/*”和“*/”之间的内容，执行 PHP 代码，输出“实体类”字符串。



多行注释语法对于根据代码生成文档的情况尤其有用，因为这样可以很明确地区分出各个注释，如果使用单行语法很难做到如此方便。

1.8 项目案例：自定义 Apache 的主目录

熟悉 IIS 的读者应该对虚拟目录不陌生，它可以将本地磁盘上的任何一个目录映射到 IIS 上，并通过一个别名来访问该目录上的内容。

在实际使用 PHP 开发项目时，如果每次都需要将项目文件复制到 Apache 的 htdocs 目录下不仅烦琐而且还很容易出错。其实，Apache 也提供了类似 IIS 虚拟目录的功能，这样开发人员可以将工作目录映射到 Apache，省去每次复制的麻烦，实现即时修改、即时浏览的功能。

【实例分析】

首先要创建虚拟目录对应的真实目录，假设在本实例中我们开发所使用的目录为 D:\MyPHP，希望用别名 MyPHP 映射到此目录。具体步骤如下：

（1）进入 Apache 安装目录下的 conf 子目录，用记事本打开 httpd.conf 文件。

（2）httpd.conf 是 Apache 的配置文件，在文件中利用查找功能搜索“<Directory />”关键字。

（3）找到后，将光标定位到“<Directory>”节点的结束标记“</Directory>”之后。

（4）添加如下的一行代码，指定将“D:/MyPHP”目录以别名 MyPHP 映射在 Apache 的根目录下。

```
Alias /MyPHP "D:/MyPHP"
```



在指定路径时使用的是 D:/MyPHP，而不是平时使用的 D:\MyPHP，这是 Apache 配置文件的命名规则，而非错误。

（5）使用“<Directory>”节点来指定“d:/MyPHP”目录所拥有的权限，代码如下所示：

```
<Directory "d:/MyPHP">
  Options Indexes FollowSymLinks
  AllowOverride None
  Order allow,deny
  Allow from all
```

```
    </Directory>
```

（6）保存对 httpd.conf 的修改，并重新启动 Apache 服务器。

（7）现在来验证 MyPHP 虚拟目录是否可访问。方法是将 1.2.2 节创建的 hello.php 复制到“D:\MyPHP”目录下，然后通过 http://localhost/MyPHP/hello.php 地址来访问，如果出现如图 1-31 所示的界面说明正常。

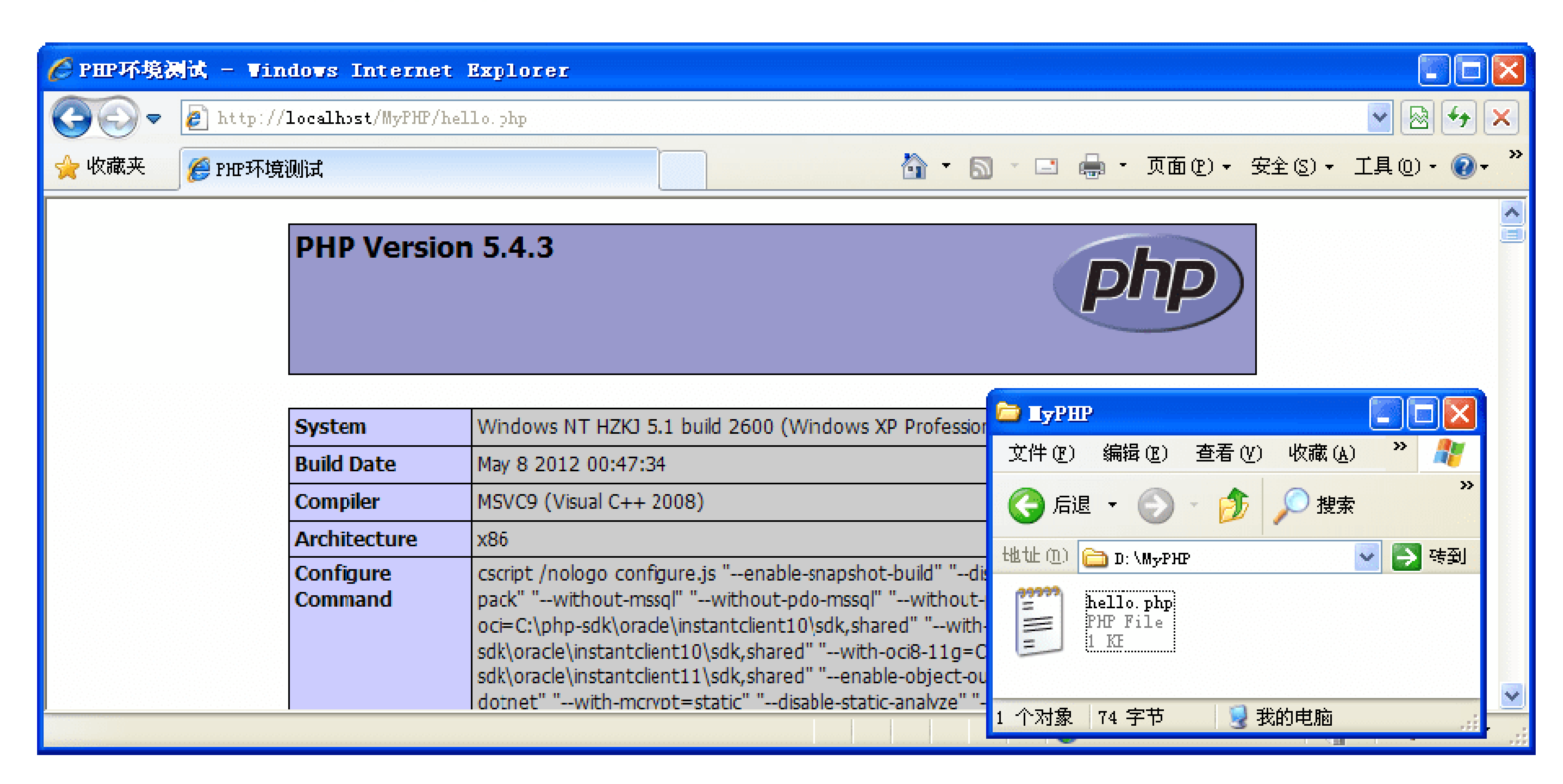


图 1-31　测试 MyPHP 虚拟目录是否可用

1.9　项目案例：在 IIS 上配置 PHP 环境

IIS 是 Windows 上自带的 Web 服务器，但是 IIS 中并没有内置对 PHP 语言的支持，因此如果需要使用 PHP，必须自行安装。PHP 可以安装为 CGI 模式或者 ISAPI 模式，由于 ISAPI 模式具有更高的性能，因此建议大家使用 ISAPI 模式。

下面以 ISAPI 模式在 Windows 2003 下的 IIS 6.0 上进行配置为例进行介绍。

PHP 具有 PHP4、PHP5 两种版本，不同的版本安装时有些区别，在此分别进行介绍。

在 PHP 官方网站（http://www.php.net/downloads.php）上提供了 PHP 解析器的两种安装包的下载链接，分别是完整文件的压缩包和不包含扩展库文件的 Installer 安装包。Installer 安装包只能安装 PHP 为 CGI 模式，如果只需要让 PHP 工作在 CGI 模式，可以下载 Installer 安装包并执行，它可以自动安装并配置 PHP。

PHP4 的安装比较简单，在此仅介绍 ISAPI 模式的安装。本文写作时 PHP4 的最新版本是 4.4.2，下载完整安装包后，将它解压到 C 盘根目录下，将其目录改名为 PHP（按个人习惯）。

【实例分析】

首先需要到 PHP 的官方网站下载 PHP 的压缩包并解压，这里为 E:\php 目录。另外，还需要确保 IIS 工作正常，具体的安装步骤这里不再介绍。详解的配置步骤如下：

（1）将 E:\php 目录中的 php.ini 复制到 C:\windows 目录下。

（2）将 E:\php 目录添加到系统环境变量的 Path 中，如图 1-32 所示。

图 1-32　添加环境变量

（3）然后用记事本打开 Windows 文件夹下的 php.ini 文件，将“extension_dir =./”改成“extension_dir = "E:\php\ext"”，最后保存即可。

（4）接下来开始设置 IIS。在 IIS 的左侧展开【Web 服务扩展】节点并右击，选择【添加一个新的 Web 服务扩展】菜单项，如图 1-33 所示。

（5）接着会出现【新建 Web 服务扩展】对话框，在【扩展名】文本框中输入“.php”。然后单击【添加】按钮将 E:\php\php5isapi.dll 添加到里面去，再启用【设置扩展状态为允许】复选框，如图 1-34 所示。最后单击【确定】按钮关闭对话框。

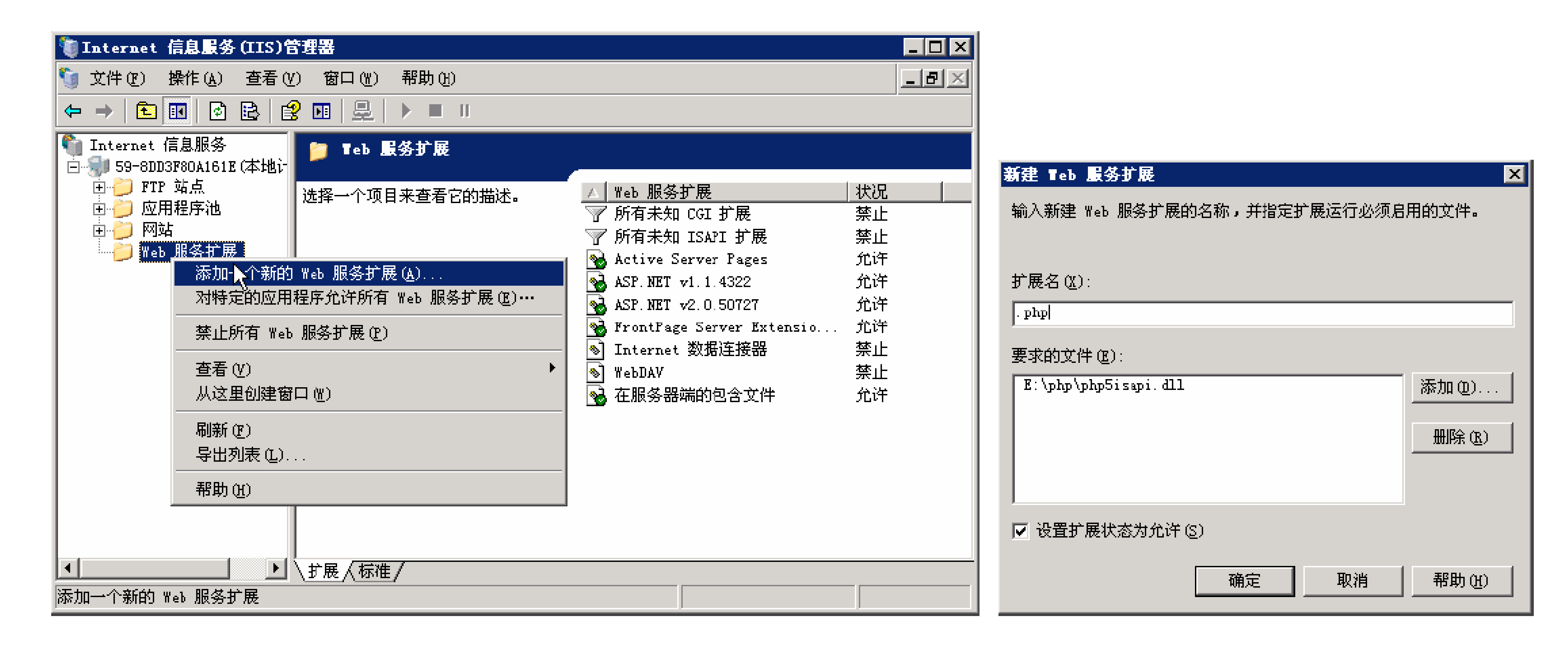


图 1-33　选择【添加一个新的 Web 服务扩展】菜单项　　　　图 1-34　设置扩展

（6）设置完毕后，还要对网站的属性进行设置。在 IIS 中打开站点的【属性】对话框，在【主目录】选项卡中单击【配置】按钮，如图 1-35 所示。

（7）在弹出的【应用程序配置】对话框中单击【添加】按钮。然后在【扩展名】文本框中输入“.php”，在【可执行文件】文本框中输入上面所提到的 E:\php\php5isapi.dll，如

图 1-36 所示。最后单击【确定】按钮关闭对话框。



图 1-35　站点的【属性】对话框



图 1-36　【应用程序配置】对话

（8）经过上面的步骤，配置过程就算完成了。下面在站点的根目录下创建一个 test.php 文件来测试 PHP 的运行环境是否搭建成功。该文件的内容很简单，如下所示：

```
<?php
phpinfo();
?>
```

（9）在浏览器中输入 http://localhost/test.php，如果能打开则说明配置正确，页面效果如图 1-37 所示。

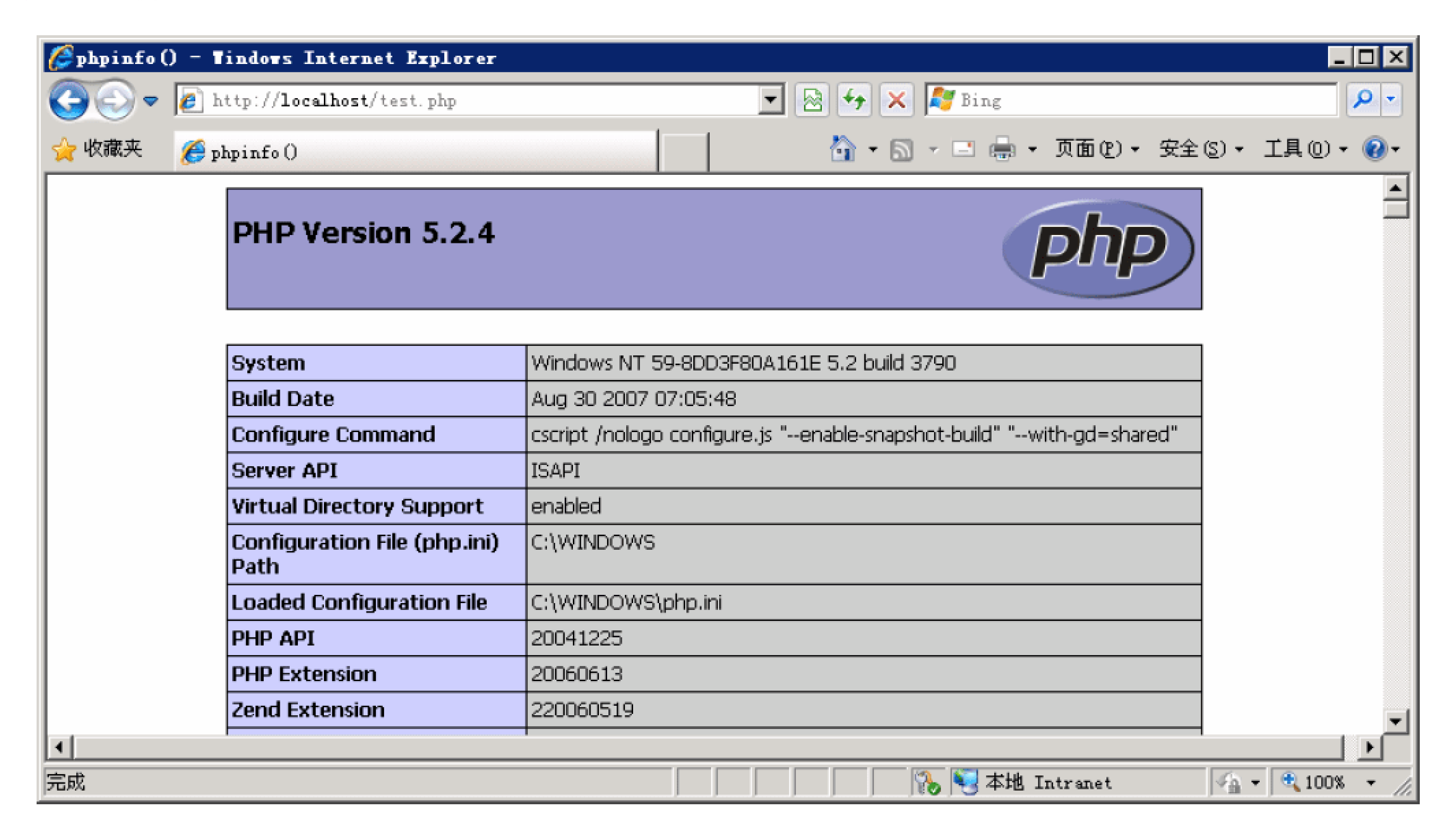


图 1-37　IIS 下运行 PHP 测试页面

到此为止，PHP 运行环境已经搭建成功了。当然，这里做的只是一些很基本的设置，在 Windows 文件夹下 php.ini 里面还有许多其他配置，读者可以自己添加。

1.10 习题

一、填空题

（1）PHP 最初的名称是__________。
（2）PHP 的配置文件__________是一个 ASCII 文本文件。
（3）PHP 程序默认以__________为开始标记，以?>为结束标记。
（4）Apache 的网站根目录是__________。
（5）假设有如下一段 PHP 代码，运行后的结果为__________。

```
<?php
    printf("(1.234 * 100) = %.2f", (1.23 * 100));
?>
```

二、选择题

（1）下面关于 PHP 的描述不正确的是__________。
A．使用内存保存经常执行的代码
B．免费、开放源代码和跨平台
C．编译后为二进制文件
D．支持面向对象
（2）启动 Apache 的命令是__________。
A．httpd
B．apache
C．net start apache
D．httpd -s
（3）下列选项中不属于安装 PHP 压缩包的是__________。
A．添加解压目录到环境变量
B．修改 Apache 配置文件
C．指定 PHP 的扩展名和初始化路径
D．指定主机地址和端口
（4）下列选项与 PHP 没有关系的是__________。
A．PHPnow
B．WampServer
C．Apache
D．Python
（5）下面代码中哪一个选项不能正确执行？__________
A．<php echo "内容" php>
B．<? echo "内容" ?>

C．<% echo "内容" %>

D．<?php echo "内容" ?>

（6）假设有如下一段 PHP 代码，运行后的输出结果为__________。

```
<?php
    #$numr=100;
    //$num=1000;
    $num=10000;
    echo ($num);
?>
```

A．100

B．1000

C．10000

D．不能执行

三、上机练习

1．安装 Zend Studio 开发工具

工欲善其事，必先利其器。开发 PHP 程序之前，选择一款好的工具能减少开发人员的很多工作，提高开发效率。

Zend Studio 是专业的 PHP 集成开发环境，它包括 PHP 所有必须的开发部件。本次上机要求读者从 http://www.zend.com 网站下载最新的 Zend Studio 版本，然后安装并编写一个 PHP 页面进行测试。

2．部署 PHP 论坛

Discuz 是目前最流行的 PHP 论坛之一，它的下载地址是 http://download.comsenz.com/DiscuzX/。本次上机要求读者下载 Discuz 程序并在 WampServer 和 PHPnow 集成环境下进行部署。

1.11　实践疑难解答

1.11.1　php.ini 不起作用的问题



php.ini 不起作用的问题

网络课堂：http://bbs.itzcn.com/thread-19189-1-1.html

【问题描述】：按照教程上的介绍，已经成功安装了 PHP。现在的问题是，如何知道 PHP 的配置文件 php.ini 已经被应用了呢？为什么对它进行了修改，却没有生效？

请问怎么解决？

【正确回答】：首先你要知道 php.ini 是 PHP 的配置文件，也是必须的。至少使用的 php.ini

是否生效，可以调用 phpinfo()函数从结果中查看 Configuration File 选项的结果。该值指定当前使用 php.ini 的完整路径，检查一下与你的是否一致。如果只显示一个目录则说明没有使用任何 php.ini 文件，应将你的 php.ini 文件放到该目录中。

UNIX 中默认在/usr/local/lib 目录中，也就是安装位置/lib。也可以在编译时通过--with-config-file-path 标记来改变路径。例如可以将路径设为：

```
--with-config-file-path=/etc
```

然后从发行包中将 php.ini-dist 复制为/etc/php.ini 并编辑它来作出想要的修改。

```
--with-config-file-scan-dir=PATH
```

Windows 中 php.ini 文件的默认路径在 Windows 目录下。如果使用的是 Apache 服务器，则会首先在 Apache 的安装目录中寻找 php.ini，例如 C:\Program Files\Apache Group\Apache。这样同一台机器上不同版本的 Apache 就可以有不同的 php.ini 文件。

1.11.2 安装成功，访问 PHP 脚本时出错



安装成功，访问 PHP 脚本时出错
网络课堂：http://bbs.itzcn.com/thread-19190-1-1.html

【问题描述】：在 Windows 系统下已经安装了 Apache 和 PHP，但是当我试着通过浏览器访问 PHP 脚本时，得到如下错误：

```
cgi error: The specified CGI application misbehaved by not returning a complete set of HTTP headers. The headers it did return are:
```

这是什么原因啊，求指教。

【正确回答】：出现这个错误信息意味着 PHP 根本就不能产生任何输出。

如果要查看更详细的错误信息，可以在命令行中转到 PHP 可执行程序（Windows 中是 php.exe）所在目录下并运行 php -i。此时如果 PHP 运行有任何问题，那么会显示相应的错误信息，这将给下一步要做什么提供线索。

如果得到满屏幕 HTML 代码（phpinfo()函数的输出），说明 PHP 本身工作正常。一旦 PHP 在命令行中工作正常，试着通过浏览器再次访问脚本。如果还失败的话那可能是如下原因。

（1）文件权限问题。你的 PHP 脚本 php.exe、php4ts.dll、php.ini 或任何要加载的 PHP 扩展库是匿名 internet 用户 ISUR_<machinename>无权访问的。

（2）脚本文件不存在（或者有可能不在你以为的地方，注意 web 文档的目录）。注意在 IIS 中通过 Internet 服务管理器设定脚本映射时选中【检查文件是否存在】选项可以捕捉到此错误。这样一来如果脚本文件不存在，服务器就会返回一个 404 错误信息。还有一个额外的好处就是 IIS 会基于 NTLanMan 权限来替你对脚本文件做任何所需要的认证。

第2章 PHP 语法快速入门

几乎每种语言都有它自己的语法，PHP 语言也不例外。要学习 PHP 程序开发，掌握其语法是必备的基础。PHP 是一种 HTML 嵌入式的脚本语言，它的语法非常灵活，混合了 C、Java、Perl 以及 PHP 自创新的语法，容易被初学者掌握。

本章详细介绍 PHP 中常量的使用、变量的命名规则、变量作用域、数据类型、类型检查和转换、各种类型的运算符以及运算符的优先规则等。通过本章的学习，无论是初学者，还是有 C、ASP 或者 Java 基础的读者都能够很快掌握 PHP 的语法。

本章学习要点：

- 掌握常量的声明和使用
- 熟悉变量的命名规则以及与常量的区别
- 掌握变量的赋值方式和作用域
- 熟悉布尔型、浮点型、整型和字符串型的定义
- 理解资源类型和空类型
- 掌握类型转换的方法和相关函数的使用
- 掌握各类运算符的使用方法
- 熟悉运算符的优先级及修改方法

2.1 常量

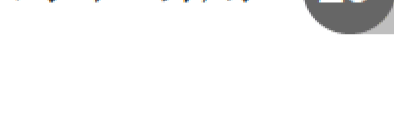
如果一个值在程序执行过程中始终保存不变，那称它为常量（Constant）。例如，圆周率的值、一年中月份的值等。

PHP 中的常量可以是表示某一数值的字符或字符串，常被用来标识、测量和比较。

2.1.1 声明和使用常量

由于 PHP 区分大小写，所以 PHP 默认约定常量的标识符总是大写，而且常量的作用范围是全局的，可以在脚本的任何地方访问。在定义和使用常量时，需要掌握以下几点。

- **常量名和其他任何 PHP 标记一样都遵循相同的命名规则，即合法的常量名以字母或下划线开始，后面跟任何字母、数字或下划线。**
- **常量标识符总是大写的。**
- **一旦定义常量，就不能再改变或者取消定义。**

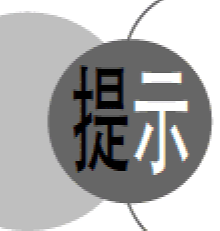


- ❑ **常量只能包含单一类型的数据，像整型或者字符串。**
- ❑ **获取常量值的时候，需要指定常量的名字，但是不需要加上$符号。**

在 PHP 中使用 define()函数定义常量，并能够赋值。该函数的语法格式如下所示：

```
boolean define(string name,mixed value [, bool case_insensitive])
```

其中 name 表示要定义的常量名称；value 表示常量的值；case_insensitive 表示在引用该常量时，是否区分大小写，该值如果为 true，表示不区分大小写。



在 PHP 5.3 以后版本中，可以使用 const 关键字在类定义的外部定义常量。一个常量一旦被定义，就不能再改变或者取消定义。

如果使用了一个未定义的常量，PHP 假定想要的是该常量本身的名字，如同用字符串调用它一样（HELLO 对应"HELLO"），同时将发出一个 E_NOTICE 级的错误。

例如，下面代码演示了如何声明常量：

```
<?php
    // 合法的常量名
    define("PI","3.1415926");
    define("MAXLENGTH","100M");
    define("TITLE","PHP 视频大全");
    // 非法的常量名
    define("2MONTH", "February");
    //下面的常量虽然是合法的，但是不推荐这种用法，因为 PHP 系统常量以  开头
    define("__FOO__", "year");
    //从 PHP 5.3 之后新增的常量命名方式
    const MIN VALUE = 0.0;
    const MAX VALUE = 1.0;
    //输出 HELLO_WORLD 字符串，并同时发出一个提示性信息
    echo HELLO WORLD;
?>
```

常量和变量的区别有以下几点。

- ❑ **常量前面没有美元符号（$）；变量需要使用$符号表示。**
- ❑ **常量只能用 define()函数定义，不能使用赋值语句；变量不需要使用 define()函数。**
- ❑ **常量可以在任何地方定义和访问；变量有它对应的作用域范围。**
- ❑ **常量一旦定义就不能被重新定义或者取消定义；变量可以重新定义。**
- ❑ **常量的值有数据类型限制；变量不受这个限制。**

【实践案例 2-1】

下面创建一个案例讲解如何定义和使用常量，具体代码如下所示：

```
<?php
   define("SKIN","清爽酷蓝");
   define("URL","www.itzcn.com");
```

```
    define("ONLINE USERS","1348");
    const NAME="财经博客";
    echo "欢迎访问 ".NAME.", 请记住我们的网址是: ".URL;
    printf("\n 当前皮肤【%s】 | 在线人数 【%s】",SKIN,ONLINE_USERS);
?>
```


在上述代码中定义了 4 个常量，并输出到页面上。执行后的输出结果如下所示：

```
欢迎访问 财经博客，请记住我们的网址是: www.itzcn.com
当前皮肤【清爽酷蓝】 | 在线人数 【1348】
```

2.1.2 系统常量

PHP 为运行时的脚本提供了大量的预定义常量，简称系统常量。这些常量中的很多都是由扩展库定义的，只有在加载了这些扩展库时才会出现。例如，使用__file__常量可以获取文件的路径和名称。表 2-1 所示为常用的系统常量。

表 2-1 PHP 系统常量

常量名	说明	注意
__file__	当前 PHP 程序文件名	file 前后都是双下划线
__line__	当前执行语句在 PHP 程序文件中的行数	line 前后都是双下划线
PHP_VERSION	当前 PHP 的版本号	PHP 必须大写
PHP_OS	当前所用操作系统类型	PHP 必须大写
E_ERROR	指明最近一次产生不可恢复的错误	大写
E_WARING	指出有错误，但程序可以继续	大写
E_PARSE	语法错误，分析器将被停止分析	大写
E_NOTICE	产生异常，但不一定是错误，程序可以继续	大写

2.2 变量

在程序中，除了常量以外，使用最多的就是变量。与常量不同，变量是指在程序运行过程中随时可以发生变化的量。它的值是临时的，当程序运行时存在，程序一旦结束，变量的值也随着丢失。

本节将详细介绍 PHP 中有关变量的内容。

2.2.1 变量的命名规则

在使用变量之前必须先创建它，而在创建时首先需要定义变量的名称。在 PHP 中变量名区分大小写，并且必须以$符号开头，然后是变量名。

变量的命名规则有如下 3 点。

- 变量名必须以字母或下划线“_”开头。
- 变量名只能包含字母、数字、字符以及下划线。

- 变量名不能包含空格。如果变量名由多个单词组成，那么应该使用下划线进行分隔（例如$my_php_root），或者以大写字母开头（例如$MyPHPRoot）。

例如，如下列出了一些变量命名的示例：

```
$helloeveryone;              //合法变量
HelloEveryOne;               //非法变量，原因：没有以$开头
$I'mJohn;                    //合法变量
$it'shere;                   //非法变量，原因：不能使用特殊符号
$60sui;                      //非法变量，原因：不能使用数字开头
$zhongguo Beijing;           //非法变量，原因：不能包含空格
$zhongguo_beijing;           //合法变量
$@itZcn.com;                 //非法变量，原因：不能使用特殊符号
$PI`2;                       //非法变量，原因：不能使用特殊符号
$2012 年;                    //合法变量，PHP 允许变量名中使用中文，但不推荐
```

2.2.2 变量赋值

由于 PHP 是一种弱类型语言，不需要显式声明变量，所以变量声明可以与赋值同时进行。在 PHP 中给变量赋值有两种方式：值赋值和引用赋值。

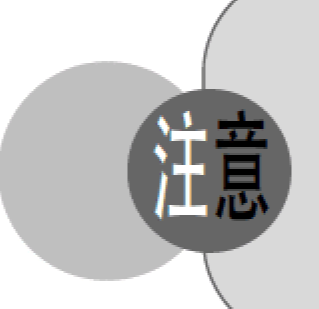
1. 值赋值

PHP 的变量默认使用传值进行赋值，它将直接把一个数值通过赋值表达式（“=”）赋值给变量。此时，该值将覆盖变量原来的值。这意味着，当一个变量的值赋予另外一个变量时，改变其中一个变量的值，将不会影响另外一个变量。

例如，下面的代码演示了这种方式的使用：

```
$bookName = "PHP 编程基础实践教程";          //字符串类型
$price=38.9;                               //浮点类型
$days=30;                                  //整型
$months=12;                                //整型
$sum=3+"15";                               //整型
$allValid=true;                            //布尔型
$bookName =$days;                          //整型
```

PHP 会在运行时检查变量类型，而且允许同一变量使用不同类型的值。例如，上述代码中的$bookName 变量，它在第一行被声明为字符串，最后一行又为它赋了一个整型数字。



虽然在 PHP 中并不需要初始化变量，但对变量进行初始化是个好习惯。未初始化的变量具有其类型的默认值，其中布尔类型的变量默认值是 false，整形和浮点型变量默认值是零，字符串型变量默认值是空字符串，数组变量的默认值是空数组。

2. 引用赋值

引用赋值是指所创建的变量与另一个变量引用的内容相同，当原始变量的内容变化时，新变量也会同时更新。使用方法为：在等号后面添加一个&符号，即“=&”，就可以完成引用赋值。

例如，下面的代码演示了这种方式的使用：

```
$city1 = "henan";                //创建一个普通变量
$city2 =&$city1;                 //使用引用方式将$city2 变量赋值为 henan
$city1="hunan";                  //为$city1 变量赋新值
echo $city1;                     //输出：hunan
echo $city2;                     //输出：hunan
$city2 = "hubei";                //为$city2 变量赋新值
echo $city1;                     //输出：hubei
echo $city2;                     //输出：hubei
```

上述代码使用引用赋值方式把$city1 的值赋给了$city2。此时如果$city1 的值发生变化，那么$city2 的值也发生变化。另外，$city2 值的变化也将影响$city1 的值。

2.3.3　可变变量

可变变量是一种特殊类型的变量，它与变量的引用方式赋值非常类似，都可以动态地设置和使用。

例如，使用如下代码创建一个值为“hello”的$str 变量。

```
$str="hello";
```

可变变量是获取一个普通变量的值作为这个可变变量的名称。例如，在上面的例子中为变量名称使用两个$符号即可创建一个可变变量。代码如下：

```
$$str="world";
```

此时，就创建了两个变量，$str 的值是“hello”，而$hello 的值是“world”。因此可以使用如下的代码来进行输出：

```
echo "$str ${$str}";
```

执行后的结果为：

```
hello world
```

由于$str 的值是 hello，所以上面的语句可以简化成如下写法：

```
echo "$str $hello";
```

2.3.4　系统变量

与系统常量一样，PHP 也提供了大量的系统变量。由于许多变量依赖于运行的服务器

的版本、设置及其他因素，所以并没有详细的说明文档。表 2-2 列出了常用的系统变量。

表 2-2　系统变量

变量名	说明
$GLOBALS	包含一个引用指向每个当前脚本的全局范围内有效的变量。该数组的键标为全局变量的名称
$_SERVER	变量由 Web 服务器设定或者直接与当前脚本的执行环境相关联
$_GET	经由 HTTP GET 方法提交至脚本的变量
$_POST	经由 HTTP POST 方法提交至脚本的变量
$_COOKIE	经由 HTTP Cookies 方法提交至脚本的变量
$_FILES	经由 HTTP POST 文件上传而提交至脚本的变量
$_ENV	执行环境提交至脚本的变量
$_REQUEST	经由 GET、POST 和 COOKIE 机制提交至脚本的变量
$_SESSION	当前注册给脚本会话的变量
$php_errormsg	前一个错误信息
$HTTP_RAW_POST_DATA	原生 POST 数据
$http_response_header	HTTP 响应头
$argc	传递给脚本的参数数目
$argv	传递给脚本的参数数组

2.3.5　变量作用域

变量作用域是指在变量声明之后它的有效作用范围。在 PHP 中变量的作用域与声明位置有关，根据声明位置可以将变量分为四类：局部变量、全局变量、函数变量和静态变量。

1. 局部变量

局部变量的作用域与它声明的位置有关，并且只在指定的范围内有效。例如，在函数内声明的变量，作用域为整个函数；在类中声明的变量，作用域为整个类；如果超出函数或者类的范围就不能访问该变量，并且不可见。

【实践案例 2-2】

例如，下面的代码演示了在函数内局部变量的使用：

```
<?php
$filename="春天花会开.mp3";
function play()
{
    $filename="传奇.mp3";
    echo "正在播放: ".$filename;       //使用函数内声明的$filename 变量
}
play();
echo "\n 正在播放: ".$filename;        //使用主程序内声明的$filename 变量
?>
```

执行后的输出结果如下所示：

```
正在播放：传奇.mp3
正在播放：春天花会开.mp3
```

可以看到，输出了两个不同的值，这是因为在 play()函数中的$filename 变量为局部变量，修改局部变量的值不会影响函数外部的任何值。$filename 变量的值在函数执行结束时被抛弃，所以主程序内$filename 变量的值没有发生变化。

2. 全局变量

与局部变量不同，全局变量的作用域最大，可以在整个 PHP 程序中的任何地方访问。声明全局变量的关键字为 global，需要在函数内才能使用。

【实践案例 2-3】

例如，下面的代码使用 global 关键字演示了全局变量的用法：

```
<?php
function UpdateMoney(){
    global $money1,$money2,$newmoney;
                        //将$money1,$money2,$newmoney 声明为全局有效
    $newmoney= $money1-$money2;
}
$money1=500;
$money2=200;
$newmoney=0;
UpdateMoney();
echo "现在的余额为：".$newmoney;
?>
```

执行后的输出结果如下所示：

```
现在的余额为：300
```

如果不在$newmoney 前加 global，该变量会被认为是局部变量，此时页面上显示的值为 0。添加 global 后，UpdateMoney()函数对变量的修改将在全局内有效，所以输出为 300。



在使用全局变量时一定要注意，因为一旦任何一个地方修改了全局变量的值，全局变量的值就发生了改变，这样很容易导致数据被意外修改。

3. 静态变量

静态变量的作用域比函数变量和局部变量大，它在声明变量时需要使用 static 关键字。与全局变量一样，静态变量仅可以用在函数内，它的值在函数退出时不会丢失，并且再次调用此函数时，还能保留值。

【实践案例 2-4】

例如，下面的代码创建了一个函数，并使用静态变量实现计数器的功能：

```
<?php
$times=10;
function getPassword()
{
    static $times=3;                        //将$times 变量声明为静态变量
    if($times<1) $times=0;
    echo "当前找回密码功能还可以使用 ".$times." 次\n";
    $times--;
}
echo "第一次调用\n";
getPassword();
echo "\n\n 第二次调用\n";
getPassword();
echo "\n 主程序内\$$times 变量的值是: ".$times;
?>
```

执行后的输出结果如下所示：

```
第一次调用
当前找回密码功能还可以使用 3 次

第二次调用
当前找回密码功能还可以使用 2 次

主程序内$number 变量的值是: 10
```

由于在 getPassword()函数中指定$times 为静态变量。因此，每次调用时都在原来值的基础上进行递减。而在函数外声明的$times 变量不会发生变化。

【实践案例 2-5】

利用静态变量在函数内有效的特性，还可以实现递归函数功能。下面是一个简单的递归函数，实现输出 10 以内的奇数。

```
<?php
function test()
{
    static $count = 1;             //使用静态变量声明初始值
    echo $count;                   //输出
    $count+=2;
    if ($count < 10) {             //判断是否超出范围
        test();
    }
}
 test();                           //调用 test()函数
```

```
?>
```

4．函数变量

函数变量指的是在创建函数时，放在函数名后面括号内声明的变量。函数变量只在函数内部有效，退出函数之后就无法访问这些变量。

【实践案例 2-6】

下面创建了一个 UserLogin()函数，它在输出时调用$username 参数的值：

```
<?php
$username="somboy";
function UserLogin($username) {
    echo "当前用户: ".$username."\n";
}
UserLogin("administrator");
echo "当前用户: ".$username."\n";
?>
```

执行后的输出结果如下所示：

```
当前用户: administrator
当前用户: somboy
```



函数参数也可以称为局部变量，因为这些参数只在函数内部起作用，在函数的外部不能访问。

2.3　数据类型

变量是用来存储数据的，它们都有一个名称和一个值。数据类型决定了如何将这些值存储到计算机的内存中。

PHP 按照数据类型值的范围可以分为标量数据类型、复合数据类型和特殊数据类型，常见的字符串和整型都属于标量数据类型。

2.3.1　标量数据类型

标量数据类型只能包含单一数据类型的数据，例如包含了字符串类型的数据，就不能再包含数字。PHP 支持四种标量数据类型：布尔型、浮点型、整型和字符串型。

1．布尔型

布尔型表示数据的真或者假，只有 true 和 false 两个值。这两个值不区分大小写，也可

以用 0 表示 false，非 0 值表示 true。

例如，下面是一些布尔型数据的示例代码：

```
$isValid= 0;                //表示 false
$canWrite = false;          //表示 false
$UpdateFrequent = -10;      //表示 true
$mySteps=5;                 //表示 true
```

2. 整型

整型就是一个不包含小数部分的数。整型可以用十进制、十六进制或八进制指定，并且前面还可以加上“+”号或者“-”号。如果用八进制符号，数字前必须加上 0，用十六进制符号，数字前必须加上 0x。

例如，下面是一些布尔型数据的示例代码：

```
$qq= 873033910;             //十进制数
$price=+59;                 //十进制数
$fuMoney = -2011500;        //一个负数
$one = 0123;                //八进制数（等于十进制的 83）
$two = 0x22;                //十六进制数（等于十进制的 34）
```



PHP 所能支持的最大整数与平台有关，一般是$\pm 2^{31}$。如果超过了此限制，将自动转换为浮点数。

3. 浮点型

浮点型数据也称为单精度数、双精度数或实数，可以指定包含小数部分的数。浮点数适用于表示货币值、重量、距离，以及用简单的整型无法满足要求的其他数据。

例如，下面是一些浮点型数据的示例代码：

```
$weight=0.518;
$length =98.1012;
$population=1312345974574.1;
$math1 =12.3e3;
$math2 = 6E-7890;
```

4. 字符串型

PHP 中的字符串是很多字符的组合，或者说是一个连续的字符数组。

要创建一个字符串，最简单的方法是用单引号（'）括起来，在字符串中用反斜线（\）转义表示一个单引号。和很多其他语言一样，也可以使用双引号（"）表示字符串，这时要注意其中的变量名会被变量值替代。

例如，下面是一些字符串数据的示例代码：

```
<?php
echo '今天天气真好啊。';                // 输出：今天天气真好啊。
echo "我们去爬山吧。";                  // 输出：我们去爬山吧。
echo 'that\'s very good';              // 输出：that's very good
echo "she's a teacher";                // 输出：she's a teacher
echo "快捷方式位于 D:\桌面收藏\游戏.exe";// 输出：快捷方式位于 D:\桌面收藏\游戏.exe
$birthday="8 月 8 日";
echo '$birthday 是我的生日';            // 输出：$birthday 是我的生日
echo "我的幸运日是今年$birthday";       // 输出：我的幸运日是今年 8 月 8 日
?>
```

如果在单引号之前或字符串结尾需要出现一个反斜线，需要用两个反斜线表示。注意如果试图转义任何其他字符，反斜线本身也会被显示出来！所以通常不需要转义反斜线本身。如表 2-3 列出了 PHP 支持的转义字符表。

表 2-3　转义字符表

序列	含义	序列	含义
\n	换行	\\	反斜线
\r	回车	\$	美元符号
\t	水平制表符	\"	双引号

2.3.2　复合数据类型

与标量数据类型相比，复合数据类型可以把多种相同类型的信息组合在一起。PHP 支持两种复合数据类型：数组和对象。由于本书后面将详细介绍它们，这里仅简单说明一下。

1. 数组类型

数组是具有相同数据类型的项集合，它们按照特定的方式进行排列和引用。例如，可以在一个数组中放置多个整数值。在 PHP 中数组里面的值按顺序排列，可以通过数组的键名（keys）加上数组名称来获得。

例如，下面的代码演示了数组的创建：

```
<?php
//使用索引创建数组
$books[0]="轻松学 PHP 编程";
$books[1]="PHP 网络大讲堂";
$books[5]="PHP 完全学习手册";
//使用键创建数组
$bands["lenovo"]="联想";
$bands["byd"]="比亚迪";
$bands["icbc"]="中国工商银行";
$bands["itzcn"]="窗内网";
?>
```

2. 对象类型

要创建一个对象首先要创建一个类，类需要用 class 关键字定义。创建好类后便可以使用 new 关键字实例化类的对象，然后再访问对象的属性、方法和成员等。

例如，下面的代码演示了类的创建及对象的使用：

```
<?php
class Book {
    var $name="PHP 编程基础与实践教程";
    var $price=38.9;
    var $pub="清华大学出版社";
}
$php=new Book();
echo "当前正在阅读由$php->pub 出版的 $php->name";
echo "\n 本书订价为 $php->price 元";
?>
```

执行结果如下所示：

```
当前正在阅读由清华大学出版社 出版的 PHP 编程基础与实践教程
本书订价为 38.9 元
```

2.3.3 特殊数据类型

除了上面介绍的两大类数据类型外，PHP 还支持一些用在特殊方面的数据类型。这些数据类型主要提供某种特殊用途，包括资源类型和空类型。

1. 资源类型

资源是一种特殊的变量，保存了一个到外部资源的引用。例如，数据库、文件和网路流等。资源是通过专门的函数来建立和使用的，它们将保持对资源的引用，直到通信结束才撤销。

当不使用资源时，资源将作为垃圾被收集起来。资源类型的变量并不真正保存一个值，实际上只保存一个指针，指向所打开的资源类型的变量，在使用时直接调用指针即可，如果输出其内容，将看到一个资源 ID 号的引用。

2. 空类型

空类型的变量只有一个 NULL 值，仅仅表示没有任何值，不表示空格，也不表示零。一个变量在下列几种情况下，会被认为是 NULL 值。

- 被赋值为 NULL。
- 没有被赋值。
- 使用 unset()函数清除。



空类型的值不区分大小写，因此 NULL 和 null 是等效的。

【实践案例 2-7】

例如，下面创建一个案例演示如何使用空类型以及测试一个变量是否为空。代码如下所示：

```
<?php
$str;
if(is_null($str))
{
    echo "变量\$str 是空\n";
}
$str=null;
if(is_null($str))
{
    echo "变量\$str 还是空\n";
}
$str=NULL;
if(is_null($str))
{
    echo "变量\$str 还是空";
}
?>
```

执行结果如下所示：

```
变量$str 是空
变量$str 还是空
变量$str 还是空
```

2.3.4 类型自动转换

当变量需要在不同的数据类型间操作时，需要转换为同一种类型，这一过程称为类型转换。PHP 支持自动转换和强制转换两种方式。

自动转换是指由 PHP 根据变量所处的环境，自动将变量转换为最适合的类型。例如，下面的示例说明自动类型转换的使用情况，具体代码如下所示：

```
<?php
$x="50.2 厘米";
$y=10;
$result=$x*$y;
echo '$x*$y='.$result;
$str="北京";
```

```
if($str)
{
echo "\n$str 是一个美丽的城市";
}
?>
```

执行结果如下所示：

```
$x*$y=502
北京 是一个美丽的城市
```

从执行结果中可以看出，变量$result 的值是由整型$y 和字符串类型$x 相乘计算的结果。由于乘法运算要求两个操作数都是数字，所以这里的$x 会自动转换为数字型。

$str 变量是一个字符串型变量，但是当用到条件语句中时就会自动转换为布尔类型，其值为 true，因此看到了输出语句。

2.3.5 类型强制转换

强制类型转换是指显式地把一种数据类型强制地转换成另外一种数据类型。经过转换的变量，将以新的类型进行计算。转换的格式是在要转换的变量前面加上要转换的类型，其形式如表 2-4 所示。

表 2-4 转换运算符

转换运算符	转换结果
(array)	数组类型
(bool)或(boolean)	布尔值类型
(int)或(integer)	整数类型
(object)	对象类型
(real)或(double)或(float)	浮点数类型
(string)	字符串类型
(array)	数组

当从数据类型范围大的向数据类型范围小的转换时，不能进行自动转换，而必须使用强制类型转换。例如，把一个数从浮点类型转换为整型，会损失小数部分。

强制转换的示例代码如下：

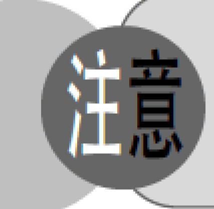
```
<?php
$number1=99.93;
$number2=(int)$number1;
echo '$number1 转换为整型后是: '.$number2;

$str="PHP 开发";
$toint=(int)$str;
echo "\n 字符串转整型后是: ".$toint;
$toobject=(object)$str;
```

```
echo "\n 字符串转对象后是: ".$toobject->scalar."\n";
$city[0]='北京';
echo var dump((boolean)$city['0']);
echo var dump((boolean)$city[1]);
?>
```


执行结果如下所示：

```
$number1 转换为整型后是: 99
字符串转整型后是: 0
字符串转对象后是: PHP 开发
bool(true)
bool(false)
```



任何一种数据类型都可以转换为对象，转换后就成为该对象的一个属性，属性名为 scalar，可以通过对象名引用该属性。

2.3.6　与类型有关的函数

除了强制类型转之外，PHP 还提供用于完成类型之间转换功能的函数。例如，使用 gettype()函数获取变量的数据类型，使用 settype()函数将变量转换为指定类型等。

1．gettype()函数

gettype()函数用于获取指定变量的数据类型，并以字符串形式返回。语法格式如下所示：

```
string gettype ( mixed $var )
```

可以看到，该函数只有一个参数，表示指定要获取数据类型的变量。gettype()函数的使用方法也很简单，例如下面的示例代码。

```
<?php
$money=2050.9;
echo "\$money 的类型是:".gettype($money);
$size="20 厘米*20 厘米";
echo "\$size 的类型是:".gettype($size);
$ary[0]="王牌";
echo "\$ary 的类型是:".gettype($ary);
echo "\$ary[0]的类型是:".gettype($ary[0]);
?>
```

执行结果如下所示：

```
$money 的类型是:double
$size 的类型是:string
```

```
$ary 的类型是:array
$ary[0]的类型是:string
```

2. settype()函数

settype()函数可以将变量强制转换为指定的数据类型，语法格式如下所示：

```
bool settype ( mixed $var, string $type )
```

如果转换成功将返回 true，否则返回 false。例如，下面的示例演示了它的用法：

```
<?php
$long="9.08 千米";
echo "\$long 的类型是:".gettype($long);
settype($long, "integer");
echo "转换后\$long 的类型是:".gettype($long);
echo "转换后\$long 的值是:".$long;
$cn="-85";
if(settype($cn, "int"))
{
    echo "\$cn 变量成功由字符串转为整型，现在的值是: ".$cn;
}
?>
```

执行结果如下所示：

```
$long 的类型是:string
转换后$long 的类型是:integer
转换后$long 的值是:9
$cn 变量成功由字符串转为整型，现在的值是: -85
```

3. 类型检测函数

PHP 的类型检测函数有很多，使用它们可以检测变量是不是属于指定的数据类型。例如，前面使用的 is_null()函数就属于此类，它用于检测变量是不是为空。

最常用的类型检测函数有：is_array()、is_bool()、is_float()、is_int()、is_null()、is_numeric()、is_object()、is_resource()、is_scalar()和 is_string()。这些函数的语法格式相同，并且返回值全部是布尔值，这里以 is_string()函数为例讲解，语法格式如下所示：

```
bool is_string ( mixed $var )
```

该函数返回的是布尔类型，如果为 string 类型则返回 true，否则返回 false，其他函数跟此函数意义基本相同。

【实践案例 2-8】

下面创建一个案例讲解如何使用 is_string()函数检查变量的数据类型，代码如下所示：

```
<?php
```

```
$ver=0.12345;
if(is string($ver))
{
    echo "\$ver 的数据类型是字符串";
}else
{
    echo "\$ver 的数据类型是".gettype($ver);
    settype($ver, "string");
    echo "\n 经过 settype()函数转换后\$ver 的数据类型是".gettype($ver);
    echo "\n 现在的值是: ".$ver;
}
?>
```

执行结果如下所示：

```
$ver 的数据类型是 double
经过 settype()函数转换后$ver 的数据类型是 string
现在的值是: 0.12345
```

2.4　运算符

运算符用于指定在程序中执行何种计算，通常针对一个以上操作数来进行运算。例如：2+3 中的操作数是 2 和 3，而运算符则是“＋”用于执行求和。还有前面多次出现的“=”也是一种运算符，它可以将一个值赋予指定变量。

PHP 支持大多数编程语言中的运算符，包括：算术运算符、赋值运算符、逻辑运算符和比较运算符等。这些运算符按照操作数的数量可以分为：一元运算符、二元运算符和三元运算符。

2.4.1　赋值运算符

赋值运算符是使用频率最高的运算符之一，其中最基本的赋值运算符是“=”。赋值运算符就是将右表达式的值存储在左操作数指定的存储位置。PHP 常用的赋值运算符如表 2-5 所示。

表 2-5　赋值运算符

运算符	说明	示例	结果
=	赋值	$n = 5	$n 的值为 5
+=	加法赋值	$n += 4	$n 的值为$n 加 4
*=	乘法赋值	$n*= 5	$n 的值为$n 乘以 5
/=	除法赋值	$n /= 10	$n 的值为$n 除以 10
-=	减法赋值	$n -= 2	$n 的值为$n 减 2
%=	取模赋值	$n %= 2	$n 的值为$n 取模 2

例如，下面代码演示如何使用这些赋值运算符：

```
<?php
$width=39;                    //结果$width=39
$height=($small=2)*10;        //结果$height=20, $small=2
$width+=1;                    //结果$width=40
$small*=5;                    //结果$small=10
$height-=4;                   //结果$height=16
$width/=5;                    //结果$width=8
$height%=3;                   //结果$height=1
$str="Hello ";                //结果$str="Hello "
$str.="PHP";                  //结果$str="Hello PHP"
?>
```

2.4.2 字符串运算符

字符串运算符有两个，第一个是连接运算符（“.”），它返回其左右参数连接后的字符串。第二个是连接赋值运算符（“.=”），它将右边参数附加到左边的参数后，作为一个新的值再赋值给左边变量。

例如，下面代码演示如何使用字符串运算符：

```
<?php
$white="白色";
$str="黑色 ".$white." 红色";
echo $str."\n";;
$str.=" 绿色";
echo $str."\n";
$white.=" 英文为 white ";
echo $white;
?>
```

执行后的输出如下：

```
黑色 白色 红色
黑色 白色 红色 绿色
白色 英文为 white
```

2.4.3 算术运算符

算术运算符是最基本的运算符，PHP 中算术运算符包括+（加法）、-（减法）、*（乘法）、/（除法）、%（取模）和-（取反）。如表 2-6 中列出了他们的用法。

例如，下面代码演示如何使用算术运算符。

```
<?php
$n=93;
```

表 2-6　算术运算符

运算符	名称	示例	结果
-	取反	$n=-50	$n 的值为-50
+	加法	$n=10+5	$n 的值为 15
-	减法	$n=10-5	$n 的值为 5
*	乘法	$n=10*5	$n 的值为 50
/	除法	$n=7/3	$n 的值为 2.333
%	取模	$n=7%3	$n 的值为 1

```
$n1=$n+7;echo $n1."\n";          //结果为: 100
$n2=$n1/5;echo $n2."\n";         //结果为: 20
$n3=$n1%6;echo $n3."\n";         //结果为: 4
$n4=$n1-12;echo $n4."\n";        //结果为: 88
$n5=$n2*2;echo $n5."\n";         //结果为: 40
$n6=-$n4;echo $n6."\n";          //结果为: -88
$n7=$n1/3;echo $n7."\n";         //结果为: 33.333333333333
$n8=$n1%3;echo $n8."\n";         //结果为: 1
?>
```

PHP 中的除法运算符“/”总是返回浮点数。除非两个操作数都是整数（或字符串转换成的整数）并且正好能整除，这时它返回一个整数。另外，取模运算符“%”的操作数在运算之前都会转换成整数（除去小数部分）。

2.4.4　递增和递减运算符

递增和递减运算符是一元运算符，可以使某一个变量自动加 1 或者减 1。这些运算符如表 2-7 所示。

表 2-7　递增和递减运算符

运算符	名称	示例	效果
++	递增运算符	++$num1	$num1 的值加 1，然后返回$num1
		$num1++	返回$num1，然后将$num1 的值加 1
--	递减运算符	--$num1	$num1 的值减 1，然后返回$num1
		$num1--	返回$num1，然后将$num1 的值减 1

使用递增和递减运算符时，要注意运算符表达式的值以及使用运算符之后变量的值。下面代码演示如何使用自增自减运算符：

```
<?php
$n=5;
$n1=$n++;          //结果$n=6, $n1=5
$n2=++$n;          //结果$n=7, $n2=7
$n2+=$n++;         //结果$n=8, $n2=14
$n3-=--$n;         //结果$n=7, $n3=-7
$n4=$n--;          //结果$n=6, $n4=7
```

```
?>
```

2.4.5 位运算符

位运算符允许对整型数中指定的位进行置位。如果左右参数都是字符串，则位运算符将操作字符的 ASCII 值。位运算符如表 2-8 所示。

表 2-8 位运算符

运算符	名称	示例	结果
&	按位与	$x & $y	将把$x 和$y 中都为 1 的位设为 1
|	按位或	$x | $y	将把$x 或者$y 中为 1 的位设为 1
^	按位异或	$x ^ $y	将把$x 和$y 中不同的位设为 1
~	按位非	~ $x	将$x 中为 0 的位设为 1，反之亦然
<<	左移	$x << $y	将$x 中的位向左移动$y 次（每一次移动都表示"乘以 2"）
>>	右移	$x >> $y	将$x 中的位向右移动$y 次（每一次移动都表示"除以 2"）

例如，下面代码演示如何使用位运算符：

```
<?php
$val=6;
$places=1;
$res=$val|$places;
p($res,$val,"|",$places);
$val=8;
$places=7;
$res=$val^$places;
p($res,$val,"^",$places);
$val=6;
$places=2;
$res=$val>>$places;
p($res,$val,">>",$places);
$val=6;
$places=2;
$res=$val<<$places;
p($res,$val,"<<",$places);
function p($res, $val, $op, $places) {
    $format = '%0' . (PHP_INT_SIZE * 8) . "b\n";

    printf("表达式: %d = %d %s %d\n", $res, $val, $op, $places);

    echo " 十进制:\n";
    printf("  val=%d\n", $val);
    printf("  res=%d\n", $res);
```

```
        echo " 二进制:\n";
        printf('  val=' . $format, $val);
        printf('  res=' . $format, $res);
    }
    ?>
```

执行后的输出结果如下：

```
表达式: 7 = 6 | 1
 十进制:
  val=6
  res=7
 二进制:
  val=00000000000000000000000000000110
  res=00000000000000000000000000000111
表达式: 15 = 8 ^ 7
 十进制:
  val=8
  res=15
 二进制:
  val=00000000000000000000000000001000
  res=00000000000000000000000000001111
表达式: 1 = 6 >> 2
 十进制:
  val=6
  res=1
 二进制:
  val=00000000000000000000000000000110
  res=00000000000000000000000000000001
表达式: 24 = 6 << 2
 十进制:
  val=6
  res=24
 二进制:
  val=00000000000000000000000000000110
  res=00000000000000000000000000011000
```

2.4.6 逻辑运算符

逻辑运算符通常用在控制语句中，如 if 或 while 等。逻辑运算符使用的是布尔类型的操作数，返回结果为 ture 或 false。逻辑运算符如表 2-9 所示。

表 2-9 逻辑运算符

运算符	名称	示例	结果
and	逻辑与	$num1 and $num2	如果$num1 与$num2 都为 true，返回 true；否则返回 false
or	逻辑或	$num1 or $num2	如果$num1、$num2 任意一个的值为 true，返回 true；否则返回 false
xor	逻辑异或	$num1 xor $num2	如果$num1 或$num2 任意一个为 true，但不同时为 true，返回 true；否则返回 false
!	逻辑非	! $num1	如果$num1 不为 true，返回 true；否则返回 false
&&	逻辑与	$num1 && $num2	如果$num1 与$num2 都为 true，返回 true；否则返回 false
||	逻辑或	$num1 || $num2	如果$num1、$num2 任意一个为 true，返回 true；否则返回 false

如表 2-9 所示，“逻辑与”和“逻辑或”分别都有两种不同形式的运算符，and 和&&效果一样，但 and 优先级比&&低；or 和||也是一样的道理。下面以 and 和&&为例，示例代码如下：

```
<?php
$score=95;
if($score>85 and $score<96) echo "成绩优秀，注意保持";
if($score>85 && $score<96) echo "成绩优秀，注意保持";
?>
```

上面代码中的两个 if 条件都返回 true，因此运行后会看到两个“成绩优秀，注意保持”。

2.4.7 比较运算符

比较运算符，顾名思义就是比较两个数值的大小，比较完成之后，返回的值为布尔值。比较表达式通常作为控制语句的判断条件。常见比较运算符如表 2-10 所示。

表 2-10 比较运算符

运算符	名称	示例	结果
==	等于	$a== $b	如果$a 等于$b，返回 true；否则返回 false
===	全等	$a === $b	如果$a 等于$b，并且它们的类型也相同，返回 true；否则返回 false
!=	不等	$a != $b	如果$a 不等于$b，返回 true；否则返回 false
<>	不等	$a <> $b	如果$a 不等于$b，返回 true；否则返回 false
!===	非全等	$a !== $b	如果$a 不等于$b，或者它们的类型不同，返回 true；否则返回 false
<	小于	$a < $b	如果$a 小于$b，返回 true；否则返回 false
>	大于	$a > $b	如果$a 大于$b，返回 true；否则返回 false
<=	小于等于	$a <= $b	如果$a 小于或者等于$b，返回 true；否则返回 false
>=	大于等于	$a >= $b	如果$a 大于或者等于$b，返回 true；否则返回 false

在上述比较运算符中，如果比较一个整数和字符串，则字符串会被转换为整数。如果比较两个数字字符串，则作为整数比较。

例如，下面的示例代码演示了常用比较运算符的使用。

```
echo var_dump(0=="a");                //输出 bool(true)
echo var_dump("1"=="01");             //输出 bool(true)
echo var_dump(1>"10px");              //输出 bool(false)
echo var_dump("5 米"<50000);          //输出 bool(true)
echo var_dump($str==null);            //输出 bool(true)
echo var_dump("abc"=="ABC");          //输出 bool(false)
echo var_dump(array()==array());      //输出 bool(true)
```

2.4.8　条件运算符

条件运算符是唯一的三元运算符，其格式如下所示：

```
Condition?true_statement:false_statement;
```

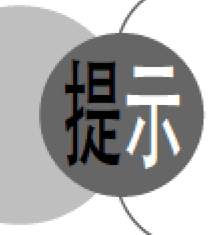
其中，Condition 表示执行的条件表达式。如果表达式的值为 true，则使用 true_statement 的值，否则使用 false_statement 的值。

例如，下面的示例代码在$x 等于$y 时，$z 的值等于 2，$x 不等于$y，那么$z 的值等于 1。

```
$z=($x==$y)?2:1;
```

下面代码可以判断$a 是否为奇数或者偶数。

```
echo  $a % 2 == 0 ?  $a."是偶数"."\n"  : $a."是奇数"."\n";
```



尽量避免同时使用多个条件运算符。因为在一条语句中使用多个条件运算符时会造成 PHP 运算结果不清晰。

2.4.9　错误控制运算符

PHP 的错误控制运算符只有一个，那就是@。把它放在一个 PHP 表达式之前，将忽略该表达式可能产生的任何错误信息。

错误控制运算符的使用示例如下：

```
<?php
    $conn=@mysql_connect("localhost","root","123456");
    echo "数据库连接已打开";
    @mysql_select_db("test");
    echo "数据库已打开";
?>
```

在使用时要注意：@运算符只对表达式有效。对新手来说一个简单的规则就是：如果能从某处得到值，就能在它前面加上@运算符。例如，可以把它放在变量、函数和 include()

调用、常量等之前，不能把它放在函数或类的定义之前，也不能用于条件结构例如 if 和 foreach 等。

2.4.10 运算符的优先规则

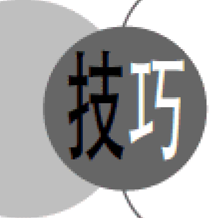
在实际的开发中，每个表达式可能由多个运算符组成。在计算时需要按照优先级的规则决定每个运算符给定表达式中的计算顺序。例如，表达式“1+2*3”的结果是 7，而不是 9。因为乘号（“*”）的优先级比加号（“+”）高。但是可以用括号来强制改变优先级。例如：“(1 + 2) * 3”的值为 9。如果运算符优先级相同，则使用从左到右的左联顺序。

在表 2-11 中从高到低列出了运算符的优先级。同一行中的运算符具有相同优先级，此时它们的结合方向决定求值顺序。

表 2-11 运算符的优先级

<table>
<tr><th>运算符</th><th>运算方向</th><th>作用</th></tr>
<tr><td>clone new</td><td>非结合</td><td>clone 和 new</td></tr>
<tr><td>[</td><td>左</td><td>array()</td></tr>
<tr><td>++ --</td><td>非结合</td><td>递增 / 递减运算符</td></tr>
<tr><td>~ - (int) (float) (string) (array) (object) (bool) @</td><td>非结合</td><td>类型</td></tr>
<tr><td>instanceof</td><td>非结合</td><td>类型</td></tr>
<tr><td>!</td><td>右结合</td><td>逻辑操作符</td></tr>
<tr><td>* / %</td><td>左</td><td>算术运算符</td></tr>
<tr><td>+ - .</td><td>左</td><td>算术运算符 和 字符串运算符</td></tr>
<tr><td><< >></td><td>左</td><td>位运算符</td></tr>
<tr><td>< <= > >= <></td><td>非结合</td><td>比较运算符</td></tr>
<tr><td>== != === !==</td><td>非结合</td><td>比较运算符</td></tr>
<tr><td>&</td><td>左</td><td>位运算符 和 引用</td></tr>
<tr><td>^</td><td>左</td><td>位运算符</td></tr>
<tr><td>|</td><td>左</td><td>位运算符</td></tr>
<tr><td>&&</td><td>左</td><td>逻辑运算符</td></tr>
<tr><td>||</td><td>左</td><td>逻辑运算符</td></tr>
<tr><td>? :</td><td>左</td><td>三元运算符</td></tr>
<tr><td>= += -= *= /= .= %= &= |= ^= <<= >>=</td><td>右</td><td>赋值运算符</td></tr>
<tr><td>and</td><td>左</td><td>逻辑运算符</td></tr>
<tr><td>xor</td><td>左</td><td>逻辑运算符</td></tr>
<tr><td>or</td><td>左</td><td>逻辑运算符</td></tr>
<tr><td>,</td><td>左</td><td>分隔表达式</td></tr>
</table>

例如，表达式：13-13%7/3 的答案是 11。这是因为在整个表达式中%的优先级最高，因此先计算 13%7，结果为 5；然后再计算 5/3，结果为 2；最后计算 13-2，最终结果为 11。



要记住这么多运算符的优先级是比较困难的，在实际应用中也不必这么清楚。因为在实际写程序时使用括号来划分优先级。

2.5 习题

一、填空题

（1）假设要定义一个名为 W，值是“www”的常量，应该使用代码__________。

（2）PHP 的变量名必须以标识符__________开始。

（3）直接与当前脚本的执行环境相关联的是__________系统变量。

（4）假如$a 的初始值为 14，分别写出下面几个变量的值。

```
$a+=$a;                 //a=
$a=12-3*2-6/4;          //a=__________
$a=(12-3)*2-6/4;        //a=__________
$a=12-3*(2-6)/4;        //a=__________
$a=(12-3)*((2-6)/4);    //a=__________
```

（5）表达式(true&&false) || (false||true)正确的结果（bool 类型的 true 或 false）__________。

（6）写出下面表达式的结果__________。

```
3 * 3 % 5
true ? 0 : true ? 1 : 2
```

二、选择题

（1）下列关于常量的描述，不正确的是__________。

A．常量有自己的命名规则，与变量不相同

B．常量定义后便不可以修改

C．可以在任何位置访问常量

D．常量前面没有美元符号（$）；变量需要使用$符号表示

（2）下面使用常量的方式，不正确的是__________。

A．define("365days","year")

B．define(YEAR,"365days")

C．echo __file__

D．const MODEL="E50.A36t1X"

（3）下列哪个变量的命名是不正确的。__________

A．$Zsconist

B．$_close_list

C．$@News_cont

D．$boud.lere

（4）假设有如下一段代码，执行后$value2 的值为__________。

```
$value1 = "one";
```

```
$value2 = &$value1;
$value1="two";
```

A. one
B. two
C. onetwo
D. oneone

（5）下列不属于比较运算符的是__________。

A. ==
B. ===
C. >=
D. ||

三、上机练习

1. 使用常量保存系统信息

常量具有全局作用域，因此可以在系统的任何位置访问。利用这个特性，本次上机要求读者创建一些常量保存商城系统的全局信息，包括网站标题、网站描述、关键字、URL、管理员用户名、管理员密码、是否开放匿名购物、是否使用验证码、同一 IP 最多注册数量、允许上传文件最大大小和缓存保存的间隔。

2. 使用变量保存商品信息

在上机实践 1 中为购物系统创建了全局信息，本次上机要求读者使用变量来模拟一件商品的信息，具体要求如下。

- 商品名称、描述和规格是字符串类型。
- 商品单价、数量、快递费用是整型。
- 是否上架，是否促销为布尔类型。
- 折扣比例是浮点类型。

3. 判断闰年

判断闰年一般的规律为:四年一闰，百年不闰，四百年再闰，其简单计算方法如下。

（1）能被 4 整除而不能被 100 整除（如 2004 年就是闰年,1800 年不是）。

（2）能被 400 整除（如 2000 年是闰年）。

根据以上描述编写一个判断闰年的表达式，并测试。

2.6 实践疑难解答

2.6.1 条件运算符计算结果的问题



条件运算符计算结果的问题
网络课堂：http://bbs.itzcn.com/thread-19684-1-1.html

【问题描述】：下面是面试时遇到的一道题目。

假设$id=0，下面输出语句的结果是什么？

```
echo $id==0?"已发送":$id==1?"未发送":"发送中";
```

我给的答案是“已发送”，可是实际测试的结果却是“未发送”。请问这是怎么回事，并解释原因？

【正确回答】：这道题目主要是测试表达式的优先级和运算方向。我们可以将它简化成如下形式：

```
a?b:c?d:e
```

在这个表达式中使用了两个条件运算符，所以具有相同优先级。此时就需要看结合性以决定运算顺序。我们知道，PHP 中条件运算符的运算方向是从左到右进行的。所以，先计算表达式 a?b:c，再计算表达式(a?b:c)?d:e，这就是所谓从左向右结合。如果是从右向左结合的话，就是先计算 c?d:e，再计算表达式 a?b:(c?d:e)。

因此问题中的表达式可以写成如下形式：

```
($id==0?"已发送":$id==1)?"未发送":"发送中";
```

括号的优先级最高先执行它，结果是“已发送”。然后表达式变成如下形式：

```
"已发送"?"未发送":"发送中"
```

在执行时会将“已发送”转换为布尔类型，结果为 true，所以最终返回“未发送”。

2.6.2 关于自增和自减运算的疑问



关于自增和自减运算的疑问

网络课堂：http://bbs.itzcn.com/thread-19685-1-1.html

【问题描述】：今天老师留了一道作业，要求将以下代码的结果输出：

```
$i=15;
$j=$i++;                    //第 1 个
echo '$i++结果'.$j."\n";
$j=++$i;                    //第 2 个
echo '++$i 结果'.$j."\n";
$j=--$i;                    //第 3 个
echo '--$i 结果'.$j."\n";
$j=$i--;                    //第 4 个
echo '$i--结果'.$j."\n";
```

程序运行结果：

```
$i++结果 15
++$i 结果 17
--$i 结果 16
```

```
$i--结果 16
```

我想知道每个结果如何得出，具体的运算思路？

【解决办法】：这道作业出得很是巧妙啊，主要考试对最简单自增和自减运算符的理解程度。

这里给你介绍一个技巧：如果变量在增减符号前（像$i++、$i--），表示先使用（赋值）后计算（加减）；相反如果变量在增减符号后（像++$i、--$i），表示先计算（加减），后使用（赋值）。

下面再来看看你的作业。

第 1 个："$j=$i++"很明显是先使用，后计算自增。那么这时$j=15，然后$i+1 变成 16，但$j 并不知道。

第 2 个：由于上面$i 已经变成 16 了，这时候"$j=++$i"先计算自增，后使用$i 的值。那么++$i 计算后为 17，因此$j 也是 17。

第 3 个：此时$i 的值是 17。"$j=--$i"先计算自减（即 17–1），再使用赋值。那么，$i 和$j 都是 16。

第四个：此时$i 的值是 16。"$j=$i--"先使用$i 的值（16）进行赋值，再将$i 进行自减（即 16–1）。结果后$j 的值是 16，而$i 的值是 15。

2.6.3　如何求表达式的值



如何求表达式的值

网络课堂：http://bbs.itzcn.com/thread-19686-1-1.html

【问题描述】：

假设有代码$a=3、$b=4、$c=5，那么下面表达式执行后的值是多少？为什么？

```
$a-3||$b+$c&&$b==$c
```

【解决办法】：这是一个典型考运算符优先级的题目。在这个表达式中算术运算的优先级最高，其次是&&运算符，最后是||运算符。

因此采用逐步分解法，表达式求值过程如下。

原始表达式：

```
$a-3||$b+$c&&$b==$c
```

将变量的值代入上面的表达式：

```
(3-3)||((4+5)&&(4==5))
=>0||((4+5)&&(4==5))
=>0||(9&&(4==5))
=>0||(9&&0)
=>0||0
=>0
```

可以看到最终结果为 0，转换为布尔类型为 false。

第3章 PHP程序流程控制

任何复杂的程序设计都离不开流程控制语句，这些语句决定了程序的走向。在所有的编程语言中，默认情况下都是按照语句的先后顺序从上到下依次执行的，这也是默认的顺序结构。为了弥补顺序结构的缺点，出现了分支结构、循环结构以及跳转结构。

本章将对组成各种结构的控制语句进行详细介绍，最后介绍 PHP 的文件引用语句。通过本章的学习，读者将对 PHP 有更多的认识，可以开发出满足更多需求的程序。

本章学习要点：

- 掌握 PHP 顺序结构的语句
- 掌握 if 语句判断各种分支的方法
- 掌握 switch 语句的使用
- 理解 if 和 switch 语句的区别
- 掌握 while 和 do while 实现循环的步骤
- 掌握 for 语句实现循环的方法
- 熟悉 return、break 和 continue 语句执行跳转的含义
- 掌握 PHP 的文件引用语句

3.1 顺序结构

前面介绍的所有程序都是按照从上到下顺序逐行执行的，也就是顺序结构。在顺序结构中流程无法跳转，因此所有语句都将执行一次。

按照语句的组成部分可以将顺序结构中的语句分为表达式语句、空语句和复合语句三大类。下面首先介绍 PHP 中语句的编写方式，然后对这三类语句进行详细介绍。

3.1.1 语句编写方式

在 PHP 中语句是最小的组成单位，每个语句必须使用分号作为结束符。除此之外，PHP 对语句无任何其他限制，开发人员可以很随意地采用符合自己风格的方式编写语句。

例如，可以将一个语句放在多行中，示例如下：

```
echo "这".
"是".
"一个".
```

```
"多行语句";
```

由于 PHP 使用分号作为语句的结束符。所以上面的三行代码会被 PHP 认为是一条语句，因为这三行中只有一行分号。但是，不推荐使用这种方式来编写语句。

同样，因为使用分号作为分隔符，将多个语句放在一行来编写也是允许的。例如，下面的示例代码也是正确的。

```
$a=1; $b=2; $c=3; $d=$a+$b*$c; printf("\$d=%s",$d);
```

在上面将 5 个语句放在一行中。

为了使程序语句排列得更加美观、容易阅读和排除错误，一般使用如下规则格式化源代码。

- **在一行内只写一个语句，并采用空格、空行来保证语句容易阅读。**
- **在每个复合语句内使用 Tab 键向右缩进。**
- **大括号总是放在单独的一行，便于检查是否匹配。**

3.1.2　表达式语句

在很多高级语言中，有专门的赋值语句。而在 PHP 中将赋值作为一个运算符，因此只有赋值表达式。在赋值表达式后面添加分号就成了独立的语句。

例如，以下是一些表达式的示例语句：

```
3.1415926;
($a+$b)/2;
$x*$y*$z-$y+(20-$x);
```

这些表达式能够被 PHP 编译器识别，但是由于没有对程序进行任何操作，因此无任何意义。

一般表达式语句应该能完成一个操作，如修改变量的值或者作为函数参数等。具体方法是，在表达式的左侧指定一个变量，来将表达式的值赋予变量；或者将表达式传递给函数等。

例如，如下是修改后的表达式语句：

```
$pi=3.1415926;
output($pi);                  //将$pi 的值传递到 output()函数中作为参数
sum=($a+$b)/2;
printf("%f",$sum);            //将$sum 的值传递到 printf()函数输出
$temp=$x*$y*$z-$y+(20-$x);    //将表达式的值保存到$temp 变量中
```

3.1.3　空语句

空语句（Empty Statement）是指在程序中什么都不做，也不包含具有实际性的语句。在程序中，空语句主要用来作为空循环体。

空语句的语法格式如下：

```
;              //其实就是一个分号
```

执行一个空语句就是将控制转到该语句的结束点。这样，如果空语句是可到达的，则空语句的结束点也是可到达的。

3.1.4 复合语句

复合语句又称为语句块，是很多个语句的组合，从而可以将多个语句看作是一个单个语句。复合语句的语法格式如下：

```
{
  statement-list  //语句列表
}
```

可以看到复合语句由一个扩在大括号内的可选 statement-list 组成。statement-list 是由一个或者多个语句组成的列表，如果不存在 statement-list，则称该语句块是空的。

它的执行规则如下。

- **如果语句块是空的，控制转到语句块的结束点。**
- **如果语句块不是空的，控制转到语句列表。当（如果）控制到达语句列表的结束点时，控制转到语句的结束点。**

【实践案例 3-1】

创建一个语句块，该语句块包含 3 条语句。

```
{
    $width = 10;                    //为$width 变量赋值
    $height = 90;                   //为$height 变量赋值
    $sum = $width + $height;        //计算$width 变量和$height 变量的和
}
```

当执行上述代码时，则$sum 变量的值为 100。上述的语句块中大括号内包含了 3 条语句。第 1 条语句为$width 变量赋值；第 2 条语句为$height 变量赋值；第 3 条语句则将$width 和$height 相加，结果保存在$sum 变量中。

3.2 分支结构

分支结构解决了顺序结构不能判断的缺点，可以根据一个条件判断执行哪些语句块。分支结构适合于带有逻辑或关系比较等条件判断的计算。例如，判断是否到下班时间、判断两个数的大小等。

在 PHP 中实现分支结构主要依靠 if 语句和 switch 语句来完成，下面详细介绍每种分支结构的具体实现。

3.2.1 单分支

在 PHP 中 if 语句是使用最多的分支语句之一，它为分支代码的执行提供了一种便利的方法。if 语句的最简单格式构成了单分支结构，此时表示“如果满足某种条件，就进行某种处理”。

语法格式如下：

```
if (表达式){
    语句块;          //满足条件时要执行的语句
}
```

如果表达式的结果为 true，就执行语句块；如果值为 false，将会忽略语句块。另外，如果语句块只有一条语句，这时可以省略大括号。单分支 if 语句的执行示意图如图 3-1 所示。

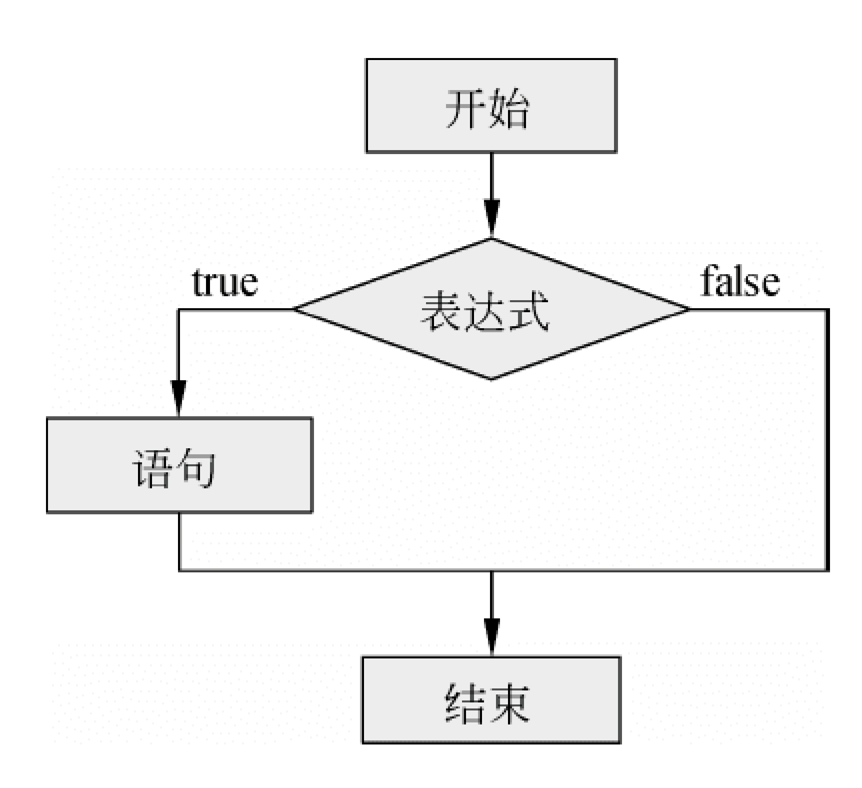


图 3-1　if 语句的执行示意图

【实践案例 3-2】

例如要比较两数的大小，希望第 1 个操作数大于第 2 个操作数时给出提示。使用 if 语句的实现代码如下所示：

```
<?php
$x=38;
$y=36;
if($x>$y)
{
    echo "\$x 大于\$y ";
    echo '$x='.$x;
}
?>
```

这里第 1 个操作数$x 的值为 38，第 2 个操作数$y 的值为 36。当执行“$x>$y”表达式时很明显返回 true，所以执行大括号内的语句之后输出“$x 大于$y $x=38”。

【实践案例 3-3】

在某一超市管理系统中，对会员有这样一个划分标准。即：消费总额达到 1000 为初级会员，消费总额达到 3000 时升级为 VIP 会员，当总额大于 10000 时就成为了高级会员。

if 的实现代码如下：

```
<?php
   if ($Consumer >= 0 && $Consumer < 3000)  echo "当前会员状态：初级会员";
   if ($Consumer >= 3000 && $Consumer < 10000)echo "当前会员状态：VIP 会员";
   if ($Consumer >= 10000)   echo "当前会员状态：高级会员";
?>
```

在上述代码中，由于每个 if 语句的语法块中只包含一条语句，所以省略了大括号号。

【实践案例 3-4】

在登录系统时要求用户名、密码和验证码都必须正确，否则将显示登录失败及错误提示。

```
<?php
$username="test";                                         //用户名
$userpass="test";                                         //密码
$code="1AC8";                                             //验证码
if($username!="admin" && $userpass!="123456" && $code!="1AC8") //比较
{
    echo "登录失败。\n";
    echo "请检查输入的用户名、密码和验证码是否正确。";
}
?>
```

在这里为 if 语句设置了一个复杂的复合表达式来验证登录条件。执行后的输出结果如下：

```
登录失败。
请检查输入的用户名和密码是否正确。
```

3.2.2 双分支

单分支结构只能对一种情况做出判断，对于其他情况无法做出相应的处理。如果在程序中，要对不符合条件的情况也做出处理，这时就可以使用双分支的 if else 语句。

if else 语句是对 if 语句的扩展，可以定义两个操作：如果条件正确，则执行一个操作，否则执行另一个操作。if else 语句的语法形式如下：

```
if (表达式){
    语句 1
}else{
    语句 2
}
```

如果表达式返回值为 true，那么执行语句 1 部分；否则将执行 else 对应的语句 2。if else 语句的执行示意图如图 3-2 所示。

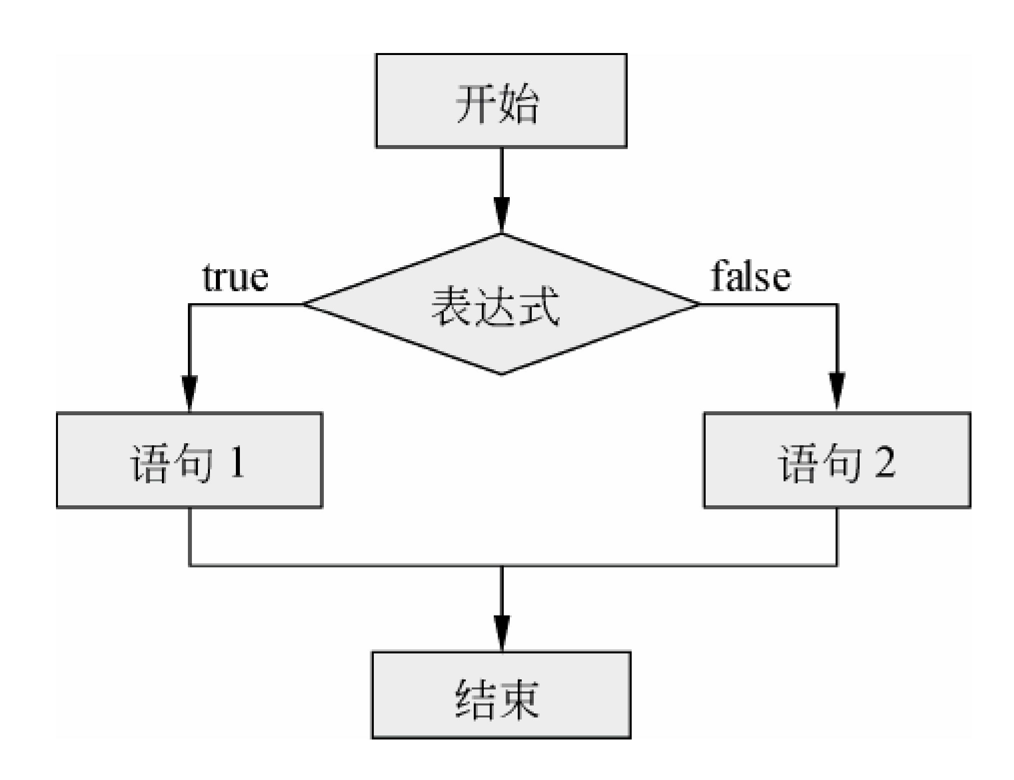


图 3-2　if else 语句的执行示意图

【实践案例 3-5】

对实践案例 3-2 进行扩展，要求无论哪个操作数大都给出提示。用双分支 if else 语句的实现代码如下：

```
<?php
$x=345;
$y=365;
if($x>$y)                          //如果$x 大于$y
{
    echo "\$x 大于\$y，最大值是：".$x;
}
else                               //否则$y 大于$x
{
    echo "\$y 大于\$x，最大值是：".$y;
}
?>
```

执行上述语句，将得到输出结果："$y 大于$x，最大值是：365"。

【实践案例 3-6】

对实践案例 3-3 进行扩展，要求无论登录成功或者失败都给出提示。使用 if else 语句的实现代码如下：

```
<?php
$username="admin";                                          //用户名
$userpass="admin";                                          //密码
$code="admin";                                              //验证码
if($username=="admin" && $userpass=="123456" && $code=="U1F4") //比较
{
    echo "欢迎用户 $username 访问系统";
}
else{
```

```
    echo "登录失败。请检查输入的用户名、密码和验证码是否正确。";
}?>
```

程序中指定了一个测试用户名、密码和验证码，当不满足 if 条件时（“$username=="admin" && $userpass=="123456" && $code=="U1F4"” 表达式返回为 false）将输出 else 后面的语句。

执行后的输出结果如下：

```
登录失败。请检查输入的用户名、密码和验证码是否正确。
```

【实践案例 3-7】

假设在考勤系统中规定 18:30 以后为下班时间。编写程序根据当前时间判断是否已经下班，并输出提示。

使用 if else 语句的实现代码如下：

```
<?php
$time="18:27";                  //定义当前时间
if($time<"18:30") echo "好好工作，努力向上";
else echo "下班时间到，请注意保存今天的工作成果。";
?>
```

【实践案例 3-8】

在银行的转账功能中，账户余额必须大于要转出的金额，否则会转账失败。这种情况下只存在两个条件，可以使用 if else 语句完成，执行代码如下。

```
<?php
$money=10000;                   //账户余额
$number=3000;                   //转账金额
if($money<$number)
{
    echo "对不起，您的账户余额不足。转账失败！";
}
else
{
    echo "转账成功。当前余额为：".($money-$number);
}
?>
```

执行后的输出结果如下：

```
转账成功。当前余额为：7000
```

3.2.3 多分支

双分支只能判断两种情况，要么符合条件，要么不符合条件。如果有多个分支情况，并且要求对每一种情况都做出反应，双分支就不能完成这样的任务。这时，可以使用多分

支 if 语句来完成，也就是说可以多个 if else 语句合并在一起使用，判断多种情况下的分支条件。

多分支条件 if 语句的语法形式如下：

```
if (表达式 1){
    语句块 1;              //满足表达式 1 时执行
} elseif (表达式 2){
    语句块 2;              //满足表达式 2 时执行
} elseif (表达式 3){
    语句块 3;              //满足表达式 3 时执行
}
……
else{
    语句块 n;              //所有条件表达式都不满足时执行
}
```

在执行时，首先判断表达式 1 的返回值，如果为 true，则执行语句块 1 部分，然后退出整个 if elseif else 语句，其他 elseif 和 else 部分都不被执行。而如果表达式 1 返回值为 false，则执行第一个 elseif，判断表达式 2 的返回值是 true 还是 false，如果为 true 则执行语句块 2，然后退出整个 if elseif else 语句，其他 elseif 和 else 部分都不被执行。如果表达式 2 返回 false，则执行第二个 elseif 判断表达式 3 的返回值，依次向下判断执行。

如果所有的 if 和 elseif 条件表达式都返回 false，那么执行 else 部分。if elseif else 语句的执行示意图如图 3-3 所示。

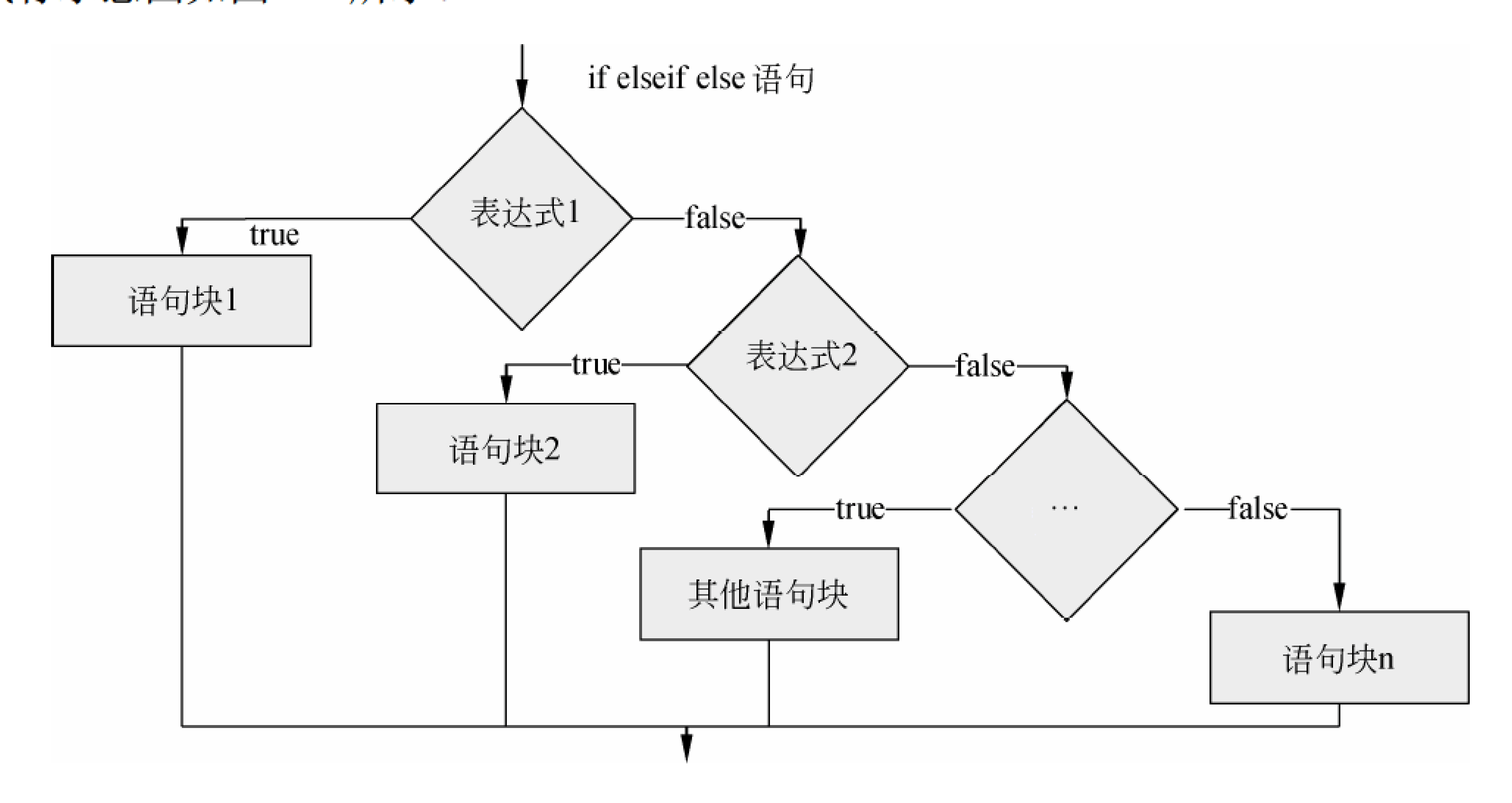


图 3-3　if elseif else 语句的执行示意图

【实践案例 3-9】

在员工绩效考核系统中将成绩分为 4 个等级：优、良、中和差。现在要实现知道等级之后，输出一个短评。使用多分支 if else 语句的代码如下：

```
<?php
```

```
$level="差";
if($level == "优")
{
    echo("成绩：优秀。\n");
    echo("评语：很不错，注意保持成绩！");
}elseif ($level == "良")
{
    echo("成绩：良好\n");
    echo("评语：继续加油!!! ");
}
elseif ($level == "中")
{
    echo("成绩：中等\n");
    echo("评语：努力!!! ");
}
else
{
    echo("成绩：差\n");
    echo("评语：要相信自己是最棒的，一定可以做的更好，努力！");
}
?>
```

如上述代码所示，虽然分为 4 个等级，但是在多分支中只需判断 3 个条件即可。如果都不满足 3 个条件，则执行 else 语句块。本实例中输出结果如下：

```
成绩：差
评语：要相信自己是最棒的，一定可以做的更好，努力！
```

【实践案例 3-10】

假设要对两个数比较大小，并输出结果提示。可能的结果为大于、小于或者等于，用多条件 if 语句的实现如下：

```
<?php
$num1=201;
$num2=202;
if($num1>$num2){
    echo "\$num1 大于\$num2，最大数为：".$num1;
}elseif($num1<$num2){
    echo "\$num2 大于\$num1，最大数为：".$num2;
}else{
    echo "\$num1 等于\$num2，都为：".$num1;
}
?>
```

如上述代码所示，$num1 和$num2 不满足 if 语句的“$num1>$num2”条件；接着测试 elseif 的“$num1<$num2”条件，满足该条件并输出“$num2 大于$num1，最大数为：202”。

【实践案例 3-11】

假设在一个支付系统中需要对用户输入的账号、密码和金额进行验证，并根据各种情况输出提示。使用 if elseif else 语句实现的代码如下：

```
<?php
define("ACCOUNT_ID", "6222 1100 0011 1234");    //用户账号
define("ACCOUNT_PASS", "000000");               //账号密码
define("ACCOUNT_MONEY", 100);                   //账号余额

$id="6222 1100 0011 1234";                      //输入的账号
$pass="000000";                                 //输入的密码
$transfer=100;                                  //要支付的金额

if($id!=ACCOUNT_ID && $pass!=ACCOUNT_PASS){
    echo "账号和密码错误";
}elseif ($id!=ACCOUNT_ID || $pass!=ACCOUNT_PASS){
    echo "账号或者密码错误";
}elseif($transfer>ACCOUNT_MONEY){
    echo "账户余额不足";
}
else {
    echo "验证成功，将从您的账户中扣除 $transfer 元";
}
?>
```

在上述代码中首先声明了 3 个常量表示账号、密码和余额，然后将用户输入的信息与它们进行对比。在比较时使用了 1 个 if 表达式和 2 个 elseif 表达式，在本实例中这些表达式都不满足，所以执行 else 语句输出结果如下：

```
验证成功，将从您的账户中扣除 100 元
```

3.2.4 分支嵌套

上面使用的 if 语句都是单层次的，其实 if 语句用法非常灵活，不仅可以单独使用还可以在 if 语句里嵌套另一个 if 语句。同样，if else 语句中也可以嵌套另一个 if else 语句。它们之间也可以互相嵌套，来完成更深层次的判断。

理论上来说，if 嵌套语句可以解决几乎所有的分支。也就是说，它可以将一系列条件表达式链接起来，对它们依次进行测试，其中一个为 true，则执行该分支。但是为了程序更容易理解，建议不要嵌套得太多，并且每个 if 语句用大括号{}括起来。

if 语句的嵌套语法格式如下：

```
if(表达式 1)
{
    if(表达式 2)
```

```
        {
            语句块 1;
        }
        else
        {
            语句块 2;
        }
    else
    {
        if(表达式 3)
        {
            语句块 3;
        }
    }
```

在上述格式中，应该注意每一条 else 与离它最近且没有其他 else 对应的 if 相搭配。例如，最简单的求 3 个数中的最大数，便可以用 if 嵌套完成，执行代码如下：

```
if (数 1 > 数 2)
{
    if (数 1 > 数 3)
    {
        数 1 是最大数
    }
    else
    {
        数 3 是最大数
    }
}
else
{
    if (数 2 > 数 3)
    {
        数 2 是最大数
    }
    else {
        数 3 是最大数
    }
}
```

【实践案例 3-12】

在安装程序时有很多步骤，任何一个步骤失败都会导致安装不成功。下面使用分支的嵌套结构来模拟安装程序，在这里验证了三项，分别是许可协议、安装位置和密钥。具体实现代码如下所示：

```
<?php
```

```
$agree=1;                                          //许可协议
$path="c:\game";                                   //安装位置
$key="芝麻开门";                                    //密钥
echo "欢迎使用本安装程序，首先请同意协议。\n";
if($agree)
{
    echo "第二步，选择安装位置\n";
    if($path!=null)
    {
        echo "第三步，输入安装密钥\n";
        if($key=="芝麻开门")
        {
            echo "正在安装....请等待\n";
            echo "安装成功。启动程序开始愉快之旅吧！";
        }else
        {
            echo "错误：请输入正确的安装密钥。";
        }
    }else
    {
        echo "错误：请指定一个有效的安装目录。";
    }
}else
{
    echo "错误：请仔细阅读安装许可协议，并同意才能继续。";
}
?>
```

在这里嵌套了 3 层，如果希望看到输出“正在安装....请等待”必须满足 3 个条件。任何一个条件不满足都将看到一项错误信息。实例的执行结果如下：

```
欢迎使用本安装程序，首先请同意协议。
第二步，选择安装位置
第三步，输入安装密钥
正在安装....请等待
安装成功。启动程序开始愉快之旅吧！
```

3.2.5 多分支的另一种实现

在前面解决分支情况时都是使用的 if 语句，它有很多语法形式。但是无论哪种形式都同时只能处理两个分支情况，类似于“二岔路口”，只能选择其中一条路来继续走。

然而生活中，经常会碰到“多岔路口”的情况。例如，在繁华的都市经常会看到米字路口，每条岔口都分别到达不同的目的地，像正前是到动物园、左边是到科技市场、右边是到汽车站等。

根据上面的描述，现在要用程序根据目的地判断往哪个方向走。很显然，用 if else 语句来描述这种“多岔路口”会显得非常麻烦，而且容易把思路打乱。

为了解决类似的问题，很多语言都提供了另一个分支语句——switch。PHP 也不例外，也可以使用 switch 语句。

switch 语句是多分支语句，适用于判断多种情况。可以把 switch 语句看作是 if elseif else 语句的另一种实现方式，如果需要比较有很多值的变量，通常会使用 switch 语句。

switch 语句的形式如下：

```
switch(表达式){
    case case1: statement1;
    case case2: statement2;
    case case3: statement3;
    ……
    default: statement4;
}
```

在执行 switch 语句时，表达式的值将分别与 case1、case2、case3 等表达式比较，根据比较结果执行对应的 statement 语句。同样的道理，首先与 case1 比较，如果与 case1 相等，执行 statement1；否则继续向下比较，如果与 case2 相等，执行 statement2。直到所有表达式都比较完了，如果还没有找到相等的，则使用 default 部分，它相当于多条件 if 语句上的 else 部分。switch 语句的执行示意图如图 3-4 所示。

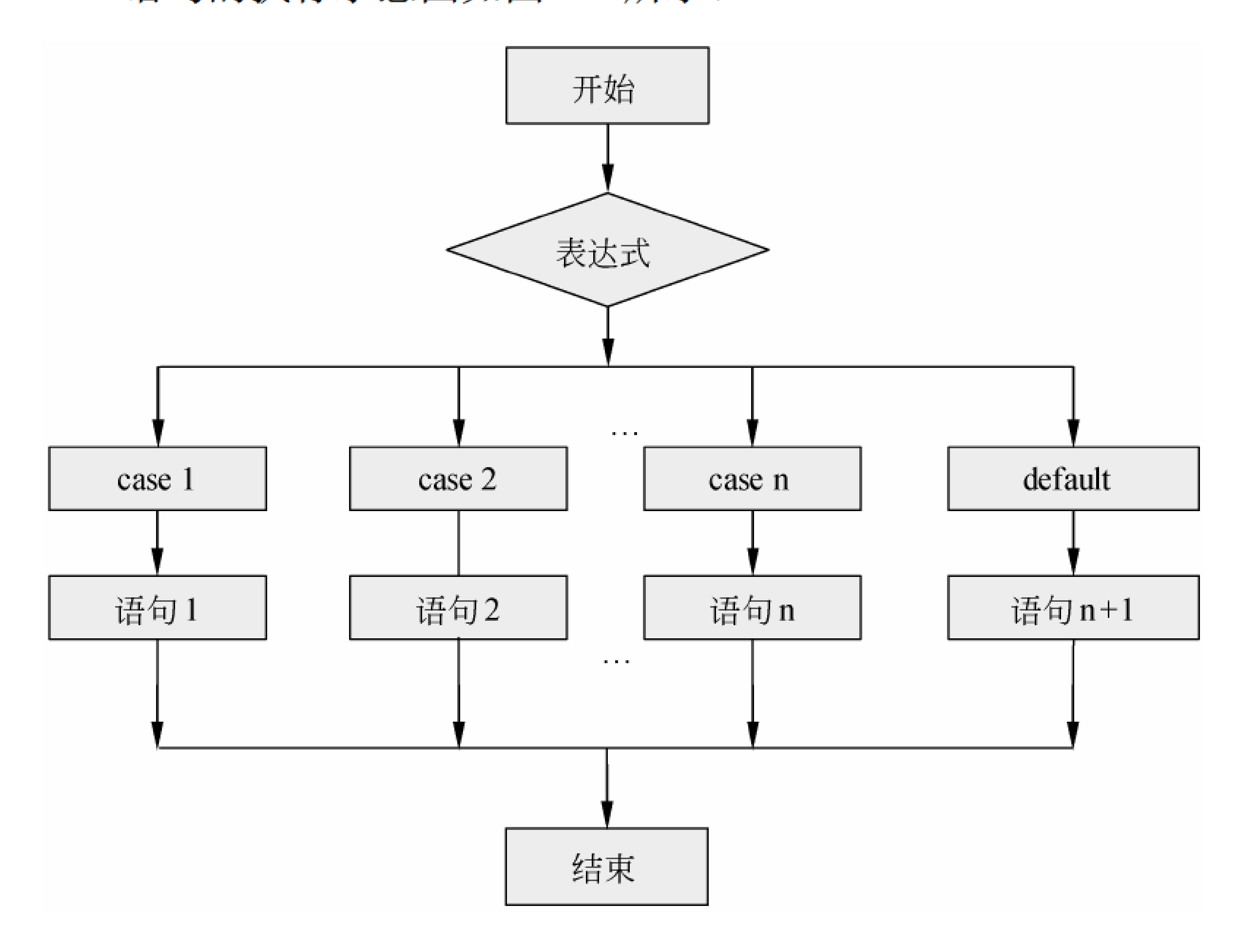


图 3-4　switch 语句的执行示意图

经过上面对 switch 语句的详细讲解之后，下面来看看如何用 switch 语句解决上面的问题，代码如下：

```
switch (目的地)
```

```
{
    case "动物园":
        往正前方向行驶
    case "科技市场":
        往左边方向行驶
    case "汽车站":
        往右边方向行驶
    case "小商品批发市场":
        往左前方向行驶
    case "建材市场":
        往右前方向行驶
    default:
        哪也不去，停车休息
}
```

【实践案例 3-13】

在一个留言本的程序中，可以对留言进行查询、更新、删除以及插入操作。现在要求使用 switch 语句来模拟这些操作，并对非法操作进行提示。实现代码如下：

```
<?php
$Action="Update";
switch ($Action)
{
    case "Select":
        echo "执行 Select 对留言信息进行查询";
        break;
    case "Update":
        echo "执行 Update 对留言信息进行更新";
        break;
    case "Delete":
        echo "执行 Delete 对留言信息进行删除";
        break;
    case "Insert":
        echo "执行 Insert 对留言信息进行插入";
        break;
default:
        echo "无匹配的操作";
}
?>
```

执行代码，结果如下所示：

```
执行 Update 对留言信息进行更新
```

【实践案例 3-14】

在一个计算器程序中，需要对操作数进行四则运算。现在要求使用 switch 语句来完成

计算，并返回计算的结果。

由题目得知，在 switch 语句中需要 4 个分支，它们的处理方式不同，下面给出了具体的实现代码：

```
<?php
function Operate($number1,$number2,$op)
{
    switch($op)
    {
        case "+":
            $result=$number1+$number2;
            break;
        case "-":
            $result=$number1-$number2;
            break;
        case "*":
            $result=$number1-$number2;
            break;
        case "/":
            $result=$number1/$number2;
            break;
    }
    echo "$number1 $op $number2= $result \n";
}
Operate(5, 2,"+");
Operate(5, 2,"/");
?>
```

由于在计算器中执行计算需要多次使用，因此这里将功能封装到 Operate()函数内。根据传递的操作数$number1 和$number2 以及执行的运算符$op 实现。switch 会将$op 与 case 分支进行匹配，以判断执行何种计算方式，最后输出结果。

```
5 + 2= 7
5 / 2= 2.5
```

3.3 循环结构

循环也是程序中的重要流程结构之一，适用于需要重复一段代码直到满足特定条件为止的情况。所有流行的编程语言中都必定有循环语句。PHP 中采用的循环语句与 C 和 Java 中的循环语句比较相似，主要有 while、do while、for 和 foreach。

3.3.1 while 语句

while 是 PHP 中实现循环的最简单语句。while 语句需要一个条件表达式以及一个循环

执行的语句块，只要表达式为 true 则一直执行语句块，直到表达式为 false 时结束。

语法形式如下：

```
while (表达式){
    语句块;                    //此处为要循环执行的语句块
}
```

执行 while 循环时，首先判断条件表达式的值，如果值为 true 则执行 while 中的语句块。执行完语句块的所有语句之后，再次判断表达式的值是否为 true，如果为 false 则退出 while 循环，如果为 true 继续执行和判断，如此循环。while 循环的执行示例图如图 3-5 所示。

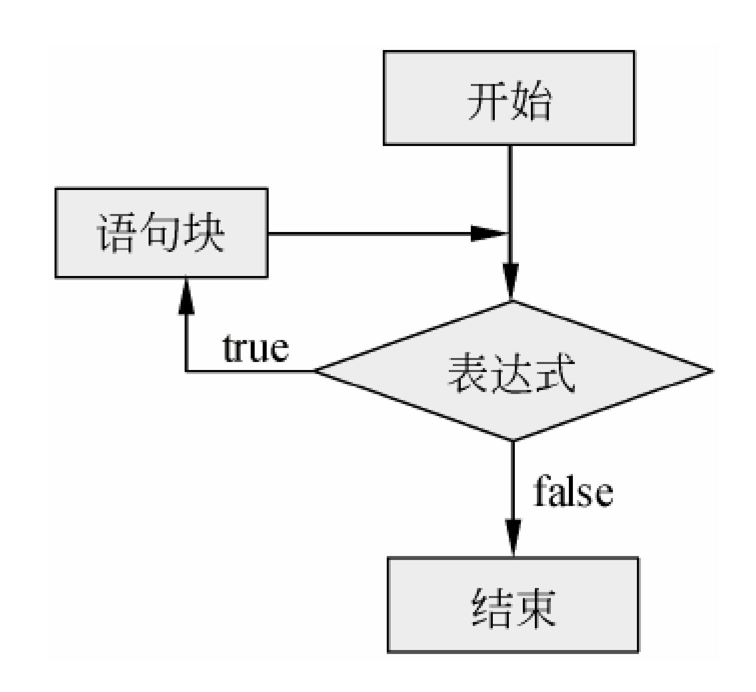


图 3-5　while 循环的执行示意图

使用 while 语句时要注意，表达式的值是在每次开始循环时检查，所以即使在语句块中该值改变了，语句也不会停止执行，直到本次循环结束。如果 while 表达式的值一开始就是 false，则循环语句一次都不会执行。

【实践案例 3-15】

编写一个程序实现输出像星期 1、星期 2……星期 7 这样的星期天数。使用 while 语句的实现代码如下：

```
$day=1;
while($day<8)                    //判断是否还要循环
{
    echo "星期$day  ";
    $day++;                      //改变$day 的值用于下一次循环
}
```

程序中$day 的初始值设为 1，将$day<8 作为循环的条件。在循环体内，输出星期字符串；然后将$day 的值递增 1，继续下一次循环。直到$number<8 条件的返回值为 false 时跳出循环。

程序执行后输出结果如下：

```
星期 1  星期 2  星期 3  星期 4  星期 5  星期 6  星期 7
```

【实践案例 3-16】

编写程序输出 1980 年至 2020 年之间的所有闰年。

实现之前首先应该知道闰年的定义，即：能被 4 整除而不能被 100 整除（如 2004 年就是闰年，1900 年不是）或者能被 400 整除（如 2000 年是闰年）。具体实现代码如下：

```
<?php
$year=1980;
while($year<2021)
{
    if(($year%4==0 && $year%100!=0) || $year%400==0)        //判断是否闰年
    {
        echo $year."年是闰年   ";
    }
    $year++;
}
?>
```

程序并不复杂，关键是 if 的条件表达式，只有为 true 时才执行 echo 输出。程序执行后输出结果如下：

```
1980 年是闰年   1984 年是闰年   1988 年是闰年   1992 年是闰年   1996 年是闰年   2000
年是闰年   2004 年是闰年   2008 年是闰年   2012 年是闰年   2016 年是闰年   2020 年是
闰年
```

【实践案例 3-17】

while 的循环语句块也可以不使用大括号{}括起来，而是使用冒号（:）和 endwhile 语句来代替。其中，冒号（:）表示循环体的开始位置，endwhile 表示 while 循环体的结束位置。

例如，下面使用这种形式实现输出 1～100 之间能够整除 13 的数，具体语句如下所示。

```
<?php
echo "1 到 100 之间能够整除 13 的数有: ";
$number=1;
while ($number<101):
    if($number%13==0)                    //判断是否能够整除 13
        echo "$number \t";               //如果能，则输出该数
    $number++;                           //改变$number 的值用于下一次循环
endwhile;
?>
```

程序执行后输出结果：

```
1 到 100 之间能够整除 13 的数有: 13   26   39   52   65   78   91
```

3.3.2 do while 语句

do while 是 while 的一种变化形式，它会先执行循环体再判断条件表达式。语法形式如下：

```
do{
    语句块;            //此处为要循环执行的语句块
}while(条件表达式);
```

如语法所示，do while 在语句块的结束处来验证循环条件表达式，而不是在开始处。这样在执行该循环时，首先执行一次语句块部分，然后判断循环表达式的值，如果值为 true 则再次执行循环；如果为 false 则退出 do while 循环。do while 循环的执行示意图如图 3-6 所示。

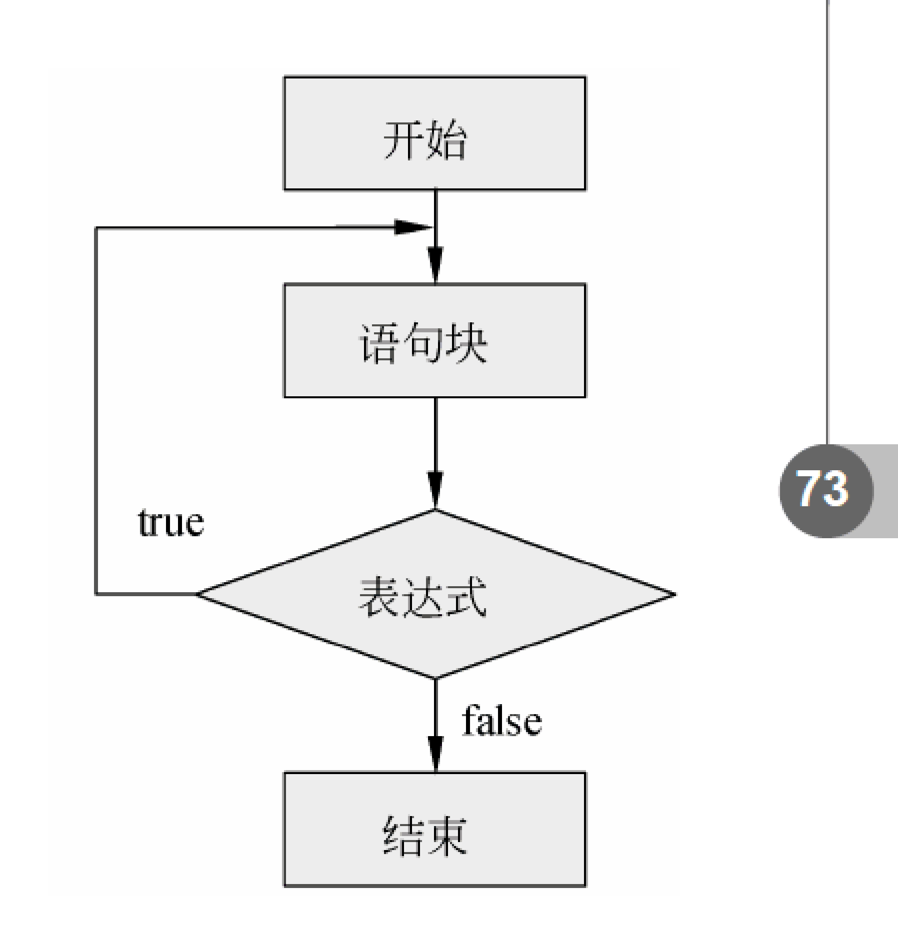


图 3-6　do while 语句的执行示意图

do while 循环与 while 循环在功能上是相似的，唯一的区别在于：while 中的语句块可能永远不会被执行，而 do while 中的语句块至少被执行一次。下面的示例说明 do while 语句的这种特性：

```
<?php
    $number = 5;
    do
    {
       echo $number;
    } while ($number > 5);
?>
```

从代码中可以看到， $number 的初始值为 5，那么条表达式返回值应该为 false，但还是会输出结果。也就是说，do while 语句不管表达式的结果是真是假，都会在执行一次后才再判断表达式的结果。代码执行后输出 5。

【实践案例 3-18】

编写一个程序实现计算从 1 到 5 的阶乘结果。使用 do while 循环的实现代码如下所示：

```
<?php
$i = 1;
$res=1;
do{
    $res*= $i;          //保存阶乘结果
    $i++;               //自增循环变量
}while($i <= 5);
echo ($i-1)."阶乘结果是: ".$res;
?>
```

程序运行后输出结果“5 阶乘结果是：120”。

3.3.3　for 语句

与 while 和 do while 不同，for 语句适合用于一个语句块重复执行预定次数的循环。因为 for 语句是在程序执行前就要判断条件表达式是否满足，如果值为 false，那么循环语句

根本就不会执行。

语法如下所示：

```
for (表达式 1; 表达式 2; 表达式 3){
    语句块;              // 循环执行的语句块
}
```

for 语句中 3 个表达式的含义如下。

- **表达式 1　初始化表达式，用于完成变量的初始化。**
- **表达式 2　循环条件表达式，值为布尔类型的表达式，表示循环条件。**
- **表达式 3　循环后操作表达式，用于调整变量的值，改变循环条件。**

在执行 for 循环时，首先执行表达式 1 完成变量的初始化工作；下一步判断表达式 2 的值，若表达式 2 的值为 true，则执行语句块部分。在执行完循环体后接着计算表达式 3，这部分通常是增加或者减少循环控制变量的一个表达式。这样一轮循环就结束了。第二轮循环从计算表达式 2 开始，若表达式 2 返回 true，则继续循环，否则跳出整个 for 循环语句。for 循环语句的执行示意图如图 3-7 所示。

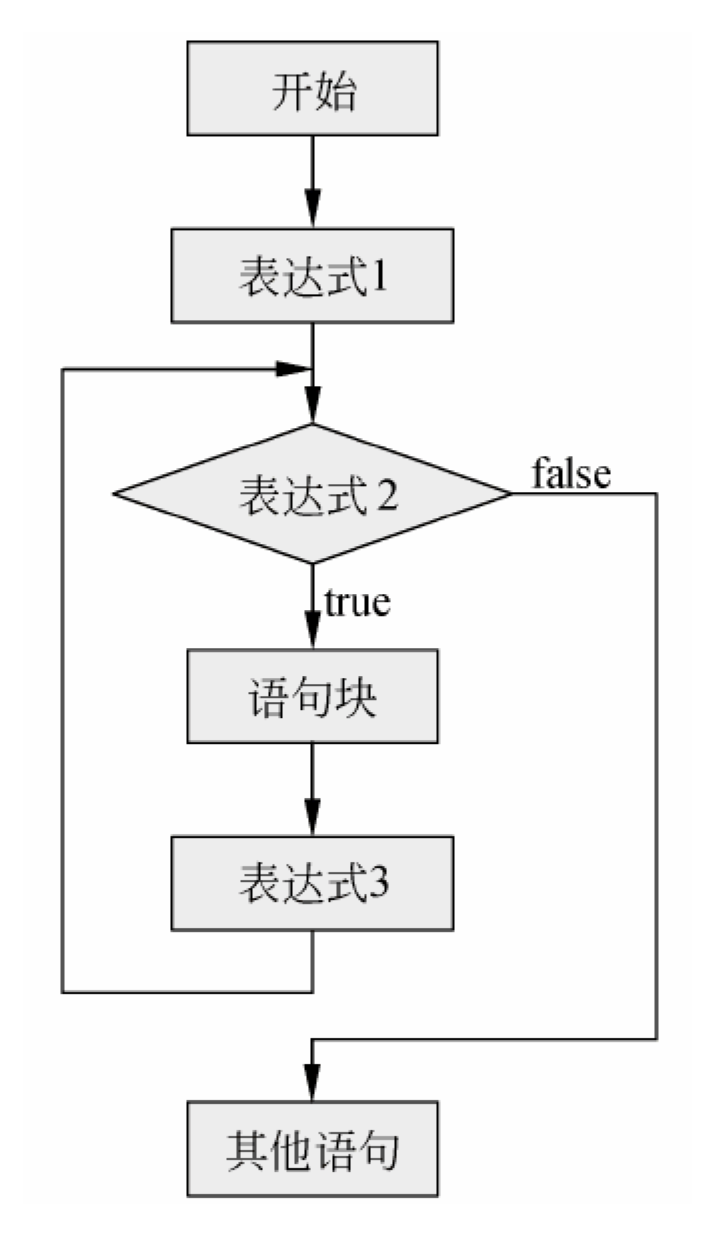


图 3-7　for 循环的执行示意图

例如，同样是计算从 1 到 5 的阶乘结果，使用 for 循环的实现代码如下：

```
<?php
$res=1;
for($num = 1; $num<= 5;$num++){
    $res *= $num;
}
echo "5 阶乘结果是: ".$res;// 输出“5 阶乘结果是: 120”
?>
```

上述语句的含义可以理解为，将$num 变量的值从 1 开始，每次递增 1，直到大于 5 时终止循环。在循环过程中，将$num 的值与当前$res 的值进行相乘。

for 语句中的 3 个表达式并不是必须存在的，它们可以部分为空，也可以全为空。下面对这些情况依次进行介绍。

1. 表达式 1 为空

for 语句表达式 1 的作用可以在程序的其他位置给出，所以当表达式 1 为空时 for 语句后面括号内其他表达式执行的顺序不变。

例如，使用 for 语句的这种形式计算从 1 到 100 之间所有奇数的和。

```
<?php
$result=0;
$number=1;                  // 相当于 for 语句的第 1 个表达式
for(;$number<101;$number++)
```

```
{
    if($number%2!=0)             //如果不能整除 2 说明是奇数则进行累加
        $result+=$number;
}
echo "100 以内所有奇数和为: ".$result;
?>
```

执行后的输出为“100 以内所有奇数和为：2500”。

2. 表达式 2 为空

当在 for 语句中表达式 2 为空时，将没有循环的终止条件。此时 for 语句会认为表达式 2 值为真，循环无限制执行下去。因此，为了使循环达到某种条件时退出，需要在语句块中进行逻辑判断，并使用 break 语句来跳出循环，否则将产生死循环。

同样是计算从 1 到 100 之间所有奇数的和，使用这种方式的代码如下。

```
<?php
$result=0;
for($number=1;;$number++)
{
    if($number>100) break;   //相当于 for 语句的表达式 2，满足时就退出 for 循环
    if($number%2!=0) $result+=$number;
}
?>
```

3. 表达式 3 为空

当在 for 语句中表达式 3 为空时，也就没有设置控制的表达式，即每次循环之后无法改变变量的值，此时也无法保证循环正常结束。

同样是计算从 1 到 100 之间所有奇数的和，使用这种方式的代码如下。

```
<?php
$result=0;
for($number=1;$number<100;)
{
    if($number%2!=0) $result+=$number;
    $number++;           //相当于 for 语句的表达式 3，每次递增 1
}
echo "100 以内所有奇数和为: ".$result;
?>
```

如果没有循环体语句，$number 变量的值为 1，永远小于 101，因此将无法结束循环，形成无限循环。而在上面代码中将$number 的递增语句放在 for 循环体内，效果与完整 for 语句相同。

4. 3 个表达式都为空

在 for 循环语句中，无论缺少哪部分表达式都可以在程序的其他位置补充，从而保持

for 循环语句完整，使循环正常进行。当 for 语句中循环体全为空时没有循环初值，不判断循环条件，循环变量不增值，此时无条件执行循环体，形成无限循环，或者死循环。对于这种情况，读者在使用时应该尽量避免。

例如，计算从 1 到 100 之间所有奇数的和，使用这种方式的代码如下。

```
<?php
$result=0;
$number=1;                          // 相当于 for 语句的表达式 1
for(;;)
{
    if($number>100) break;          // 相当于 for 语句的表达式 2
    if($number%2!=0) $result+=$number;
    $number++;                      // 相当于 for 语句的表达式 3

}
echo "100 以内所有奇数和为: ".$result;
?>
```

【实践案例 3-19】

for 语句和 if 语句相似，同样可以实现嵌套。例如，要实现输出一个同腰三角形，最终图形如下所示：

```
     *
    ***
   *****
  *******
 *********
***********
```

要输出上面的图形，我们需要这样几个循环，一个循环控制行数，一个循环控制输出空格，一个循环控制输出"*"字符。最终代码如下所示：

```
<?php
$max = 5;                       //设置三角形的高度
for($temp=0;$temp<=$max;$temp++){
    //进行空格判断当第一个时输出 6 个空格，第二个时输出 6-1=5 个空格以此类推
        for($temp2=0;$temp2<$max-$temp;$temp2++)  echo " ";
        for($temp3=0;$temp3<=$temp*2;$temp3++)  echo "*";
        echo "\n";              //输出换行符
}
?>
```

3.3.4 foreach 语句

foreach 语句主要用于执行遍历，可以从数组中提取每个“键/值”对，直到获取所有

项，或者满足某些内部条件为止。foreach 循环语句的执行示意图如图 3-8 所示。

foreach 循环语句有两种语法形式，第一种语法形式最简单，用于从数组中获取每个值，每次迭代都将指针后移。语法如下：

```
foreach (数组变量 as 遍历变量){
    语句块;
}
```

图 3-8　foreach 执行示意图

【实践案例 3-20】

假设在$keys 数组中保存了很多关键字短语。现在要求遍历该数组，并输出所有关键字。使用 foreach 语句实现代码如下所示：

```
<?php
$keys=array("php","perl","python","apache","mysql");
                                        //创建数组，并指定包含的元素
echo "热门关键字：";
foreach($keys as $key)                  //使用$key 变量遍历数组
{
    echo "$key  ";                      //输出$key 变量的值
}
?>
```

在这个例子中$keys 数组包含了 5 个元素。当使用 foreach 语句遍历时，PHP 将依次取出每个元素并赋予$key 变量，直到所有元素都使用过。在循环时将$key 变量的值输出，对于本例共循环 5 次，因此输出 5 个值。运行后输出结果如下：

```
热门关键字：php  perl  python  apache  mysql
```

foreach 循环语句的第二种形式适合处理包含键和值的数组。语法如下：

```
foreach (数组变量 as 键变量 =>值变量){
    语句块;
}
```

【实践案例 3-21】

假设在$carts 数组中保存了用户选择的商品信息（包括编号和名称），其中编号为键，名称为键值。现在要求遍历该数组，并输出所有商品的编号和名称。使用 foreach 语句实现代码如下所示：

```
<?php
$carts=array(
        "10"=>"商务男士专用钱包",
        "11"=>"夏日混搭经典衬衫",
        "15"=>"雷电无线鼠标+键盘套装",
        "28"=>"热销太阳镜[包邮]",
);                                              //保存商品信息的数组
```

```
echo "当前购物车中有如下宝贝：<br/>";
foreach ($carts as $id=>$name)                       //遍历数组
{
    echo "宝贝编号：$id    宝贝名称：$name <br/>";    //输出键名和键值
}
?>
```

在使用 foreach 语句的这种形式时，PHP 会自动查找数组并使用指定的变量替换键名和键值。在本例中，$id 保存的是数组中的键名，$name 保存的是数组中的键值。运行后输出结果如下所示：

```
当前购物车中有如下宝贝：
宝贝编号：10    宝贝名称：商务男士专用钱包
宝贝编号：11    宝贝名称：夏日混搭经典衬衫
宝贝编号：15    宝贝名称：雷电无线鼠标+键盘套装
宝贝编号：28    宝贝名称：热销太阳镜[包邮]
```

3.4 跳转结构

严格地说，跳转结构不属于流程控制结构。但是它在程序设计时也非常重要，可以帮助程序员更加精确地控制整个流程，方便程序的设计。例如在遇到死循环时使用跳转结构结束循环。

PHP 提供了与其他语言相同的跳转结构控制语句，包括 return、break 和 continue。

3.4.1 return 语句

return 语句的作用非常明确，就是用来退出循环，但是只能在函数内使用。此时不管当前循环有多少层嵌套，都会被终止，同时结束函数的运行。

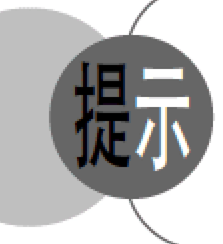
下面的示例代码演示了 return 语句的使用：

```
<?php
function TestReturn()                                  //函数开始
{
    for($number1=1;$number1<5;$number1++)
    {
        echo "第 1 层循环：$number1\n";                //return 将结束函数
        if($number1>3) return ;
        for($number2=1;$number2<5;$number2++)
        {
            echo "第 2 层循环：$number2 \n";
            if($number2>2) return ;                    //return 将结束函数
        }
    }
```

```
}
TestReturn();
?>
```

在 TestReturn()函数内使用了两层嵌套的 for 循环，无论是外层循环中的$number1>3 条件为 true，还是内层循环中的$number2>2 为 true，都将退出所有循环。执行后输出结果如下所示：

```
第 1 层循环：1
第 2 层循环：1
第 2 层循环：2
第 2 层循环：3
```



return 语句后面还可以跟一个表达式来指定函数的返回值，具体知识将在后面讨论函数时详细介绍。

3.4.2　break 语句

break 语句表示强制从当前的语句块（主要为循环语句）跳转出来，并执行语句块下面的语句。break 适用的语句有：for、foreach、while、do while 和 switch。

【实践案例 3-22】

在循环嵌套中，内部循环中的 break 关键字只能终止内部循环，而不是跳出所有循环。下面的示例代码演示了如何使用 break 语句来终止所在的循环。

```
<?php
for($i=1;$i<3;$i++)                    //外层循环总共 2 次
{
    echo "第 1 层循环：$i\n";
    for($j=1;$j<4;$j++)                //内层循环总共 3 次
    {
        echo "第 2 层循环：$j\n";
        if($j==2) break;               //但是由于 break 语句，只能循环 2 次
    }
}
?>
```

如上述代码所示，在内部循环中如果满足$j==2 就使用 break 关键字终止内部循环，但是外部循环不受影响。具体的运行结果如下所示：

```
第 1 层循环：1
第 2 层循环：1
第 2 层循环：2
第 1 层循环：2
```

```
第 2 层循环：1
第 2 层循环：2
```

从结果中可以看出，外部和内部循环都执行了 2 次。

【实践案例 3-23】

假设，要计算从数字 2 的几次幂开始大于 100。在这里使用 do while 结合 break 实现，代码如下：

```
<?php
$number=1;                          //开始数字
$result=1;                          //结果变量
do{
    $result*=2;                     //计算结果
    if($result>100) break;          //如果满足条件就退出，不再循环
    $number++;                      //不满足则自增
}while($number<100);
echo "2 的 $number 次幂结果是 $result";
?>
```

执行后输出结果为“2 的 7 次幂结果是 128”。

3.4.3 continue 语句

除了 return 语句和 break 语句外，continue 语句也可以退出循环。

continue 语句表示退出当前的循环而执行下一次循环，适用于循环语句和 switch 语句。它和 break 的区别是：使用 break 语句将退出整个循环，而 continue 语句仅仅退出当次循环，继续执行下一次循环。

【实践案例 3-24】

假设，在$users 数组中保存了很多的用户名称。现在要求输出其中的用户名称，但是由于用户 zhht 被禁用，因此在输出的结果中不希望显示它。编写代码实现输出$users 数组中的其他用户。实现代码如下所示：

```
<?php
$users=array("admin","zhht","somboy","houxia"); //保存用户名称的数组
echo "当前可用用户有: ";                        //循环开始前输出一段话
foreach ($users as $user)
{
    if($user=="zhht") continue;                 //遇到 zhht 时则不输出，继续下一次循环
    echo "$user ";
}
echo "<br/>输出结束";                           //循环结束后输出一段话
?>
```

在这里使用 foreach 来遍历$users 数组，使用其他循环语句同样可以实现。在循环体内使用 if 语句判断如果$user 的值是 zhht，则使用 continue 语句忽略下面的输出语句，继续下

一次循环。执行后输出结果如下：

```
当前可用用户有: admin somboy houxia
输出结束
```

3.5 文件引用语句

在程序编写中，为了提高代码的可重用性，通常会将一些公用的代码放到一个单独的文件中，然后根据需要使用包含语句将它们引入即可。例如数据库连接的账号、程序定义的常量、网页头部的导航栏和网页尾部的版权声明等。

PHP 提供了 4 个语句来实现文件引用，分别是 include、require、require_once 和 include_once。

3.5.1 include 和 include_once

include()语句将引用指定文件中的所有文本，并把文本复制到使用该语句所在的文件中。语法格式如下所示：

```
bool include(string $filename)
```

这里的$filename 表示要包含的文件名，如果文件已经被包含则返回 true。当一个文件被包含时，其中所包含的代码继承了 include 所在行的变量范围。从该处开始，调用文件在该行处可用的任何变量在被调用的文件中也都可用，而且所有在包含文件中定义的函数和类都具有全局作用域。

寻找包含文件的顺序是：先在当前工作目录相对的 include_path 下寻找，然后在当前运行脚本所在目录相对的 include_path 下寻找。例如，文件中有一句“include "file1.php"”，则寻找 file1.php 的顺序是先/www/，然后是/www/include/。如果文件名以“./”或者“../”开始，则只在当前工作目录相对的 include_path 下寻找。

【实践案例 3-25】

假设在网站根目录下的 config.php 文件中保存了用户的详细信息，像登录名称、密码、角色和登录时间等。具体文件内容如下：

```
<?php
 $username="Administrator";
 $userpass="*******";
 $userid=1;
 $logintimes=49;
 $role="超级管理员";
 $lasttime="2012-08-25 20:45:59";
?>
```

为了将用户信息显示到当前的 index.php 文件中，首先需要引用 config.php，然后直接

调用其中的变量。如下所示 index.php 中的这部分代码：

```
<?php include "config.php" ?>
<h1> 用户信息管理 &gt; 查询详细信息</h1>
<div style="padding:10px;font-size:14px;line-height:18px;">
    用户编号：<?=$userid?><br/>
    用户名称：<?=$username?><br/>
    登录密码：<?=$userpass?><br/>
    所属角色：<?=$role?><br/>
    登录次数：<?=$logintimes?><br/>
    上次登录时间：<?=$lasttime?><br/>
</div>
```

在这里要注意 index.php 和 config.php 位于同一个目录，运行 index.php 文件将看到如图 3-9 所示效果。

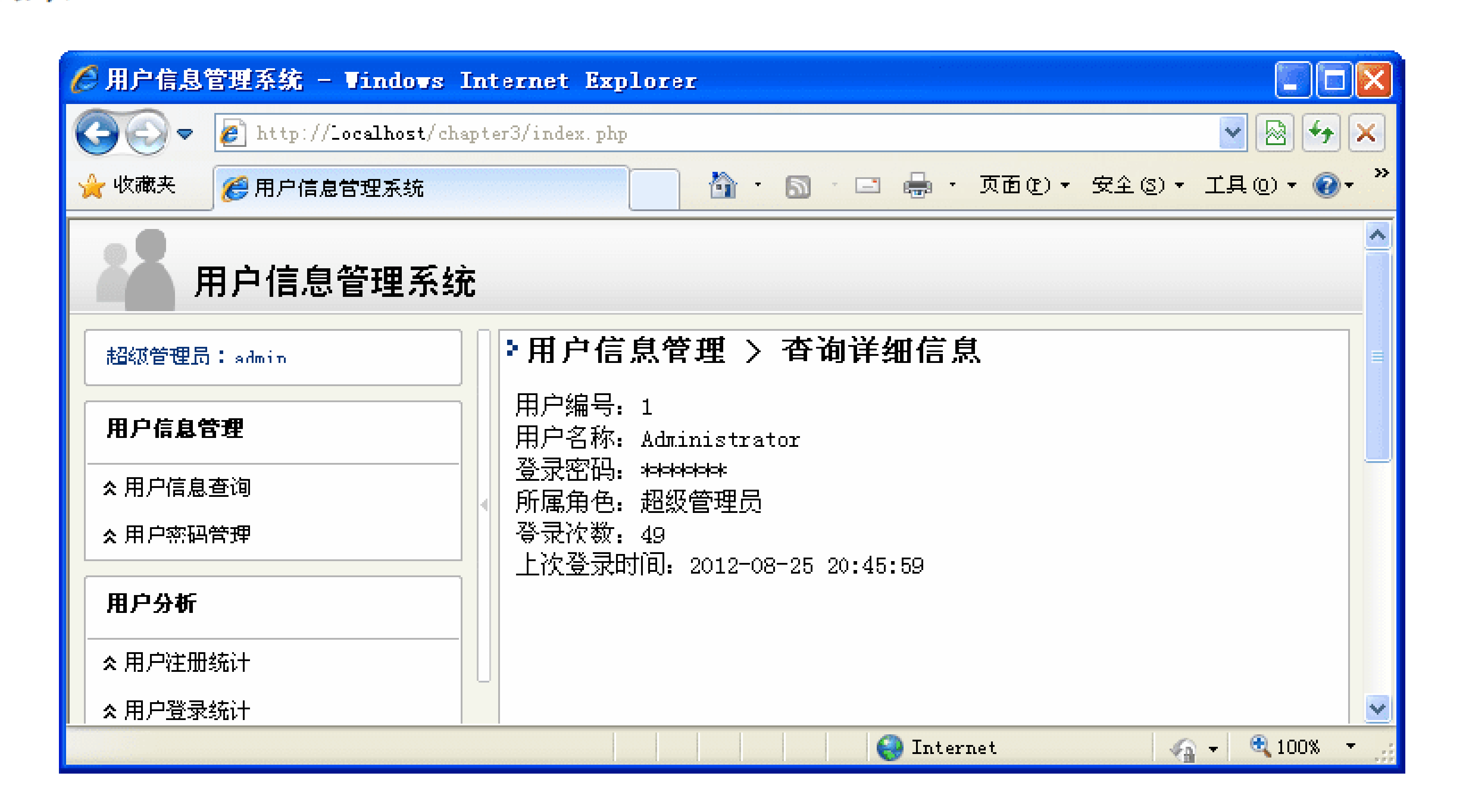


图 3-9　使用引用显示用户详细信息

include_once()语句和 include()语句类似，也用于包含文件。和 include()语句不同的是，使用 include_once()语句在包含文件时，如果该文件中的代码已经被包含了，则不会被再次包含。因此使用 include_once()语句可以确保它只被包含一次，从而避免出现函数重定义、变量重新赋值等问题。

3.5.2　require 和 require_once

require()和 include()除了处理失败的方式不同外，在各个方面都完全一样。include()产生一个警告，而 require()则导致一个致命错误。也就是说，如果想在丢失文件时停止处理页面，应该使用 require()，include()则会继续执行脚本。

【实践案例 3-26】

假设，现在使用 include()引用了不存在的文件 wrongFile.php，代码如下所示：

```
<?php
include("wrongFile.php");              //产生一个警告
```

```
echo "Hello World!";                    //此句会执行
?>
```

运行后，在页面中会得到类似下面这样的错误消息：

```
错误消息:
Warning: include(wrongFile.php) [function.include]: failed to open stream:
No such file or directory in PHPDocument2 on line 4

Warning: include() [function.include]: Failed opening 'wrongFile.php' for
inclusion (include path='d:\MyPHP') in PHPDocument2 on line 4
Hello World!
```

从上面的结果可以看到，虽然产生了错误，但是 echo 语句依然被执行了。这是因为警告不会中止脚本的执行。

现在，使用 require()函数运行相同的例子，代码如下所示：

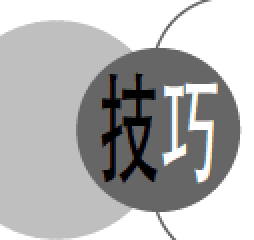
```
<?php
require("wrongFile.php");               //产生一个致命错误
echo "Hello World!";                    //此句不会执行
?>
```

运行后，在页面中会得到类似下面这样的错误消息：

```
Warning: require(wrongFile.php) [function.require]: failed to open stream:
No such file or directory in PHPDocument2 on line 4

Fatal error: require() [function.require]: Failed opening required
'wrongFile.php' (include_path=' d:\MyPHP ') in PHPDocument2 on line 4
```

由于在致命错误发生后终止了脚本的执行，因此 echo 语句不会执行。



正因为在文件不存在或被重命名后脚本不会继续执行，因此推荐使用 require()而不是 include()。

require_once()语句和 include_once()语句功能类似，它在包含文件之前也会先检查当前文件是否被包含过，如果该文件被包含了，则该文件就会被忽略，不会被再次包含。

【实践案例 3-27】

为了使读者更加深刻地理解 include_once()函数和 require_once()函数与 require()函数和 include()函数的区别，下面创建一个案例来演示，具体步骤如下。

（1）对实践案例 3-25 中的 config.php 文件进行扩展添加如下代码：

```
<?php
function check(){
    global $userid;
    if($userid==1)
```

```
        {
            echo "<b>这是网站管理员创始人的 ID，注意不能删除。</b><br/><br/>";
        }
    }
    check();
?>
```

（2）在同一目录下创建一个 test.php 文件，它的代码如下所示：

```
<?php require "config.php" ; ?>
<?php
function welcome($name)
{
    echo "<br/><b>欢迎管理员 $name 回来。</b>";
}?>
```

可以看到，在 test.php 文件中使用 require 函数引入了 config.php 文件。

（3）再创建一个 PHP 页面，在这里使用 require 函数引入上面创建的两个 PHP 文件，并输出其中的信息，代码如下所示：

```
<h1> 用户信息管理 &gt; 查询详细信息</h1>
<div style="padding:10px;font-size:14px;line-height:18px;">
<?php require "config.php" ; ?>
<?php require "test.php" ; ?>
    用户编号：<?=$userid?><br/>
    用户名称：<?=$username?><br/>
    登录密码：<?=$userpass?><br/>
    所属角色：<?=$role?><br/>
    登录次数：<?=$logintimes?><br/>
    上次登录时间：<?=$lasttime?><br/>
    <?php welcome($username)?>
</div>
```

（4）将文件保存为 require.php，在浏览器中运行会看到如图 3-10 所示效果。

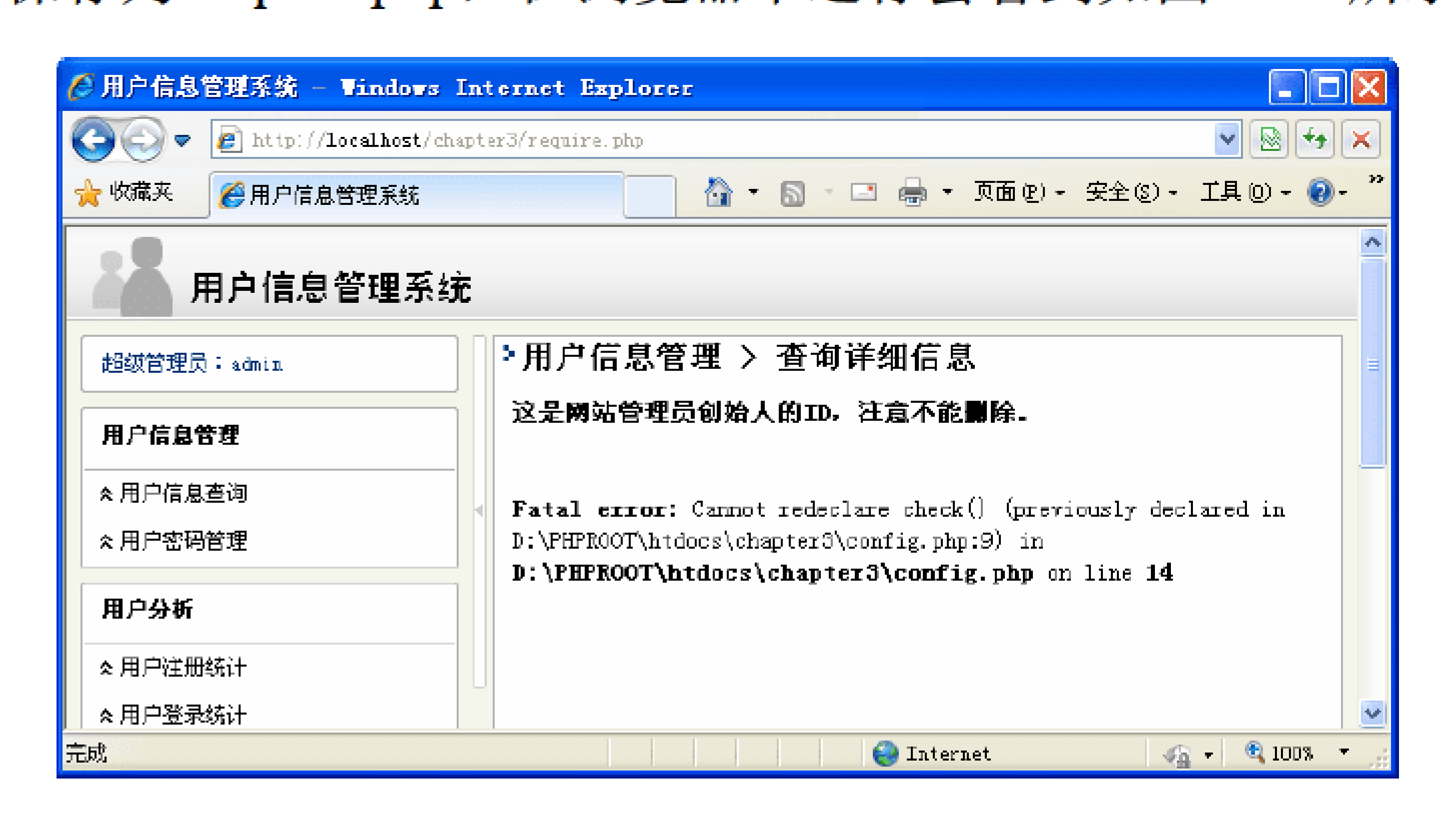


图 3-10　error_require.php 运行效果

由于在 dbTest.php 文件中已经引入了 dbConfig.php 文件。因此当运行到 error_require.php 文件的“require "dbTest.php"”语句时，dbTest.php 文件和 dbConfig.php 文件已经都引入了。所以，再往下运行到“require "dbConfig.php"”语句时会提示在 dbConfig.php 文件中声明的 TestDB()函数已经存在，无法重新定义。

（5）为了避免多次引用同一个文件导致的错误，可以使用 require_once()。具体方法是，把 require.php 文件和 dbTest.php 文件中的 require 函数换为 require_once 函数。

（6）再次通过浏览 require.php 文件查看此时的运行效果，如图 3-11 所示。

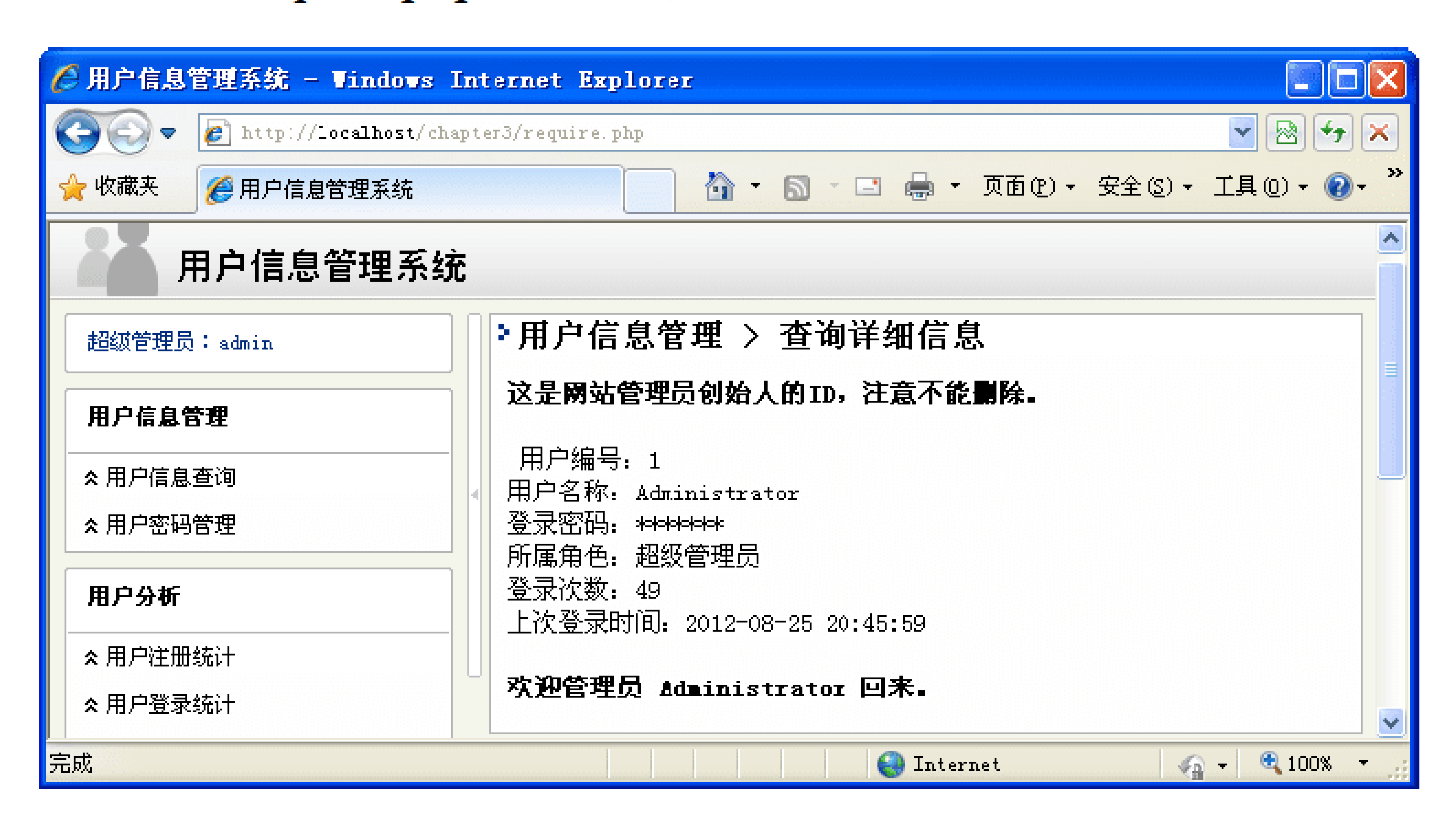


图 3-11　error_require_once.php 运行效果

从图 3-11 中可以看到，此时正确运行并输出了在 test.php 文件和 config.php 文件定义的输出。

3.6　项目案例：制作一个 PHP 网站首页

目前无论大小网站或者系统，开发者都会采用模板化技术来布局网站。这样一来，可以将网站的各个模块分散到具体的文件，然后在显示时根据需要进行组装即可；而且可以实现一处修改、网站同步更新的效果，非常快捷。

本次案例将制作一个 PHP 网站首页，将它分为三个部分，从上到下依次为页眉导航部分、页中内容显示部分和页脚版权显示部分。

【实例分析】

首先需要规划好文件的目录结构。在本实例中所有文件都位于相同目录，将头部作为 head.php，底部作为 foot.php，中间主要内容作为 index.php。然后通过文件引用语句进行包含，具体步骤如下。

（1）创建 head.php 文件，然后编写显示网站标题和导航菜单的代码。如下所示：

```
<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td height="100" background="images/header.gif"><span class="STYLE1">
    我的个人主页</span></td>
```

```
  </tr>
</table>
<table width="100%" border="0" cellspacing="1" bgcolor="006600">
  <tr align="center" valign="bottom" id="menu">
    <td><span>我的日记</span></td>
    <td><span>我的收藏</span></td>
    <td><span>我的相册</span></td>
    <td><span>我的朋友</span></td>
    <td><span>给我留言</span></td>
    <td> </td>
    <td><span class="STYLE2"></span></td>
  </tr>
</table>
```

（2）在相同目录下创建 foot.php 文件，编写显示版权信息的代码。如下所示：

```
<div class="footer">CopyRight 2012 www.itZcn.com All Rights Reserved. 窗
内网 版权所有</div>
```

（3）然后，在同一目录中创建一个 index.php 文件作为页面内容显示部分，并在这里使用文件引用语句引入 header.php 和 foot.php 文件。

```
<?php include "head.php"; ?>
<?php require "foot.php"; ?>
```

（4）在页面的主要位置编写一个使用 PHP 输出前 50 个素数的小程序。这部分布局和实现代码如下所示：

```
<table width="100%" border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td valign="top"><table width="90%" border="0" align="center"
    cellpadding="0" cellspacing="0">
        <tr>
          <td height="50" valign="middle" align="center"><b><span class=
          "STYLE8">我的第一个 PHP 小程序</span></b></td>
        </tr>
        <tr>
          <td><h3>功能：实现输出前 50 个素数</h3>
            <?php
$NUM_OF_PRIMES = 50;                    //素数数量
$count = 1;                             //起始数量
$number = 2;
$isPrime = true;                        //是否素数
while ($count<=$NUM_OF_PRIMES) {        //是否达到数量
    $isPrime=true;
    for($divisor = 2; $divisor <= $number / 2; $divisor++)
    {
```

```
            if ($number % $divisor == 0)        //判断是否能整除
            {
                $isPrime =false;
                break;
            }
        }
        if ($isPrime)
        {
            if ($count % 10 == 0)               //判断是否输出 10 个
            {
                echo "$number <br/>";           //输出素数后换行
            }
            else {                              //如果没有则直接输出
                echo "$number    ";
            }
            $count++;
        }
        $number++;
    }
    ?> <hr/>素数的定义：如果一个大于 1 的数只能被 1 或它本身整除，那么这个数就是素数。例如
    2、3、5 和 7 都是素数，而 4、6、8 和 9 都不是素数。 </td>
          </tr>
          <tr>
            <td height="50"><table width="100%" border="0" cellspacing="0"
            bgcolor="#FDFDE0">
                <tr>
                  <td width="69" height="24"><div align="center" class=
                  "STYLE49">
                      <div align="left">友情链接</div>
                    </div></td>
                  <td width="25" align="center" class="STYLE32"><a href="#">
                  百度</a></td>
                  <td width="25" align="center" class="STYLE32"><a href="#">
                  谷歌</a></span>
                    </div></td>
                  <td width="25" align="center" class="STYLE32"><a href="#">
                  新浪</a></span>
                    </div></td>
                  <td width="25" align="center" class="STYLE32"><a href="#" >
                  网易</a></span>
                    </div></td>
                  <td width="25" align="center" class="STYLE32"><a href="#">
                  搜狐</a></span>
                    </div></td>
                  <td width="45" align="center" class="STYLE32"><a href="#">
```

```
                    窗内网</a></span>
                      </div></td>
                    <td> </td>
                  </tr>
                </table></td>
            </tr>
          </table></td>
        <td width="250" align="right" valign="bottom"><img src=
        "images/tree.jpg" width="238" height="234" /></td>
      </tr>
    </table>
```

（5）对 index.php 页面进行布局的美化与调整。然后，通过浏览 index.php 来查看网站首页的运行效果，如图 3-12 所示。



图 3-12　网站首页运行效果

实现本案例的关键和开发程序都类似，主要是将问题划分为几个子问题，并分别解决每个子问题。例如，在实例中首先将网站首页拆分为多个子页面，然后对每个页面单独设计，再对首页进行实现编写输出素数的功能。

3.7　习题

一、填空题

（1）在 PHP 中使用__________语句可以将很多个语句的组合，作为单个语句来执行。

（2）如果一个语句在程序中什么都不做，也不包含具有实际性的语句，那么将它称为

__________语句。

（3）假设有下面一段 PHP 程序，执行后的输出结果是：__________。

```
$x = 11;
$y = 15;
if($x > $y)  echo " \$x 是最大数";
else  echo " \$y 是最大数";
```

（4）while 语句和 do while 语句可以实现相同的功能，其中 do while 语句的特点是__________，再判断循环条件。

（5）例如要实现输出九九乘法口诀表，下面使用 for 语句的嵌套形式实现，在空白处填写合适的代码。

```
<?php
for($i=1;$i<10;           )
{
    for($j=1; __________;$j++)
    {
        echo "$i*$j=".$i*$j;
        echo " \t";
    }
    echo "<br/>";
}
?>
```

（6）下面程序的输出结果是：__________。

```
<?php
$users=array("admin","somboy","houxia","zhht");
foreach ($users as $user)
{
    if($user=="houxia") break;
    echo "$user ";
}
?>
```

二、选择题

（1）下面语句的执行结果是__________。

```
echo "Hi".
"P".
"H".
"P";
```

A．Hi PHP

B．Hi

C．P

D．不能执行

（2）在 PHP 中，语句是最小的组成单位，下列语句不正确的是__________。

A．$a=10;

B．$b=toInt("China");

C．$a=$b=$c

D．$x*$y*$z-$y+(20-$x);

（3）下面不属于程序设计中流程结构的是__________。

A．分支结构

B．循环结构

C．顺序结构

D．跳转结构

（4）下面对 for 语句说法不正确的是__________。

A．for 语句的循环条件不能缺少

B．for(; ;)是无限循环

C．for 循环无法嵌套

D．for 语句是跳转语句

（5）下列语句的执行结果是__________。

```
$i="1";
if($i) echo "A";
if($i==1) echo "B";
if($i=="1") echo "C";
```

A．A

B．B

C．C

D．ABC

三、上机练习

1．输出菱形图案

在 3.3.3 节介绍 for 语句时实现输出一个等腰三角形的图案。现在需要读者在此基础上进行扩展，最终输出一个菱形图案，如下所示：

```
     *
    ***
   *****
  *******
 *********
***********
 *********
```

```
 *******
  *****
   ***
    *
```


2．输出水仙花数.

编写一个程序输出 1000 之内的所有水仙花数。

水仙花数是指一个 n 位数（n≥3），它的每个位上的数字的 n 次幂之和等于它本身（例如：1^3 + 5^3 + 3^3 = 153）。

3．求宰相的麦子

故事背景：相传古印度宰相达依尔是国际象棋的发明者。有一次，国王因为他的贡献要奖励他，问他想要什么。达依尔说“只要在国际象棋上摆上麦子就行了。规则是，第一格一粒，第二格两粒……后面每个格子总是前一格子麦子数的两倍。摆满整个棋盘，我就感恩不尽了。”国王一想，这还不容易，刚想答应，如果这时你在国王旁边站着，会不会劝国王别答应，为什么？

国际象棋一共 64 格，请你用编程的方式计算出摆满需要多少粒麦子。

3.8　实践疑难解答

3.8.1　使用 switch 控制范围出现的问题



使用 switch 控制范围出现的问题

网络课堂：http://bbs.itzcn.com/thread-2179-1-1.html

【问题描述】：假设，现在我有个语句是 switch($num)。现在我想判断$num 变量在哪个范围里，例如$num <0、$num>0 && num<50、$num>50 && $num<100，或者是 num>=100 等，使用 switch 行不行啊？

网上搜索了一下，说是在 VB 语言中可行，不知道在 PHP 中行不行。如果行的话，应该怎么写？如果不行的话，是不是只能用 if 来做判断了啊？

【正确回答】：很遗憾的告诉你，PHP 不允许这么做。而且这样的情况不应该使用 switch，因为 switch 是针对数值可以枚举的情况，而你的要求显然不可能枚举所有小于某个整数的值，这时应该使用 if。例如，下面的示例代码：

```
if ($num<0) {...}
elseif ($num<50) {...}
elseif ($num<100) {...}
else{...}
```

这样逻辑非常清晰，语句也非常简单。

3.8.2 PHP 中 exit、continue 和 break 的解释



PHP 中 exit、continue 和 break 的解释

网络课堂：http://bbs.itzcn.com/thread-2234-1-1.html

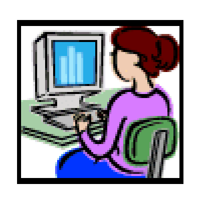
【问题描述】：我有些搞不懂在 PHP 中，if 语句中添加 continue 和 break，它们跳出循环是不是没用的，因为 if 语句根本就不是循环语句，只有在 switch、for 和 while 语句中才能跳出。

对于 exit，网上都说是终止程序，我听得云里雾里的，后来自己做了一下，原来是终止这个 exit 函数后面的那些程序，在它之后的程序全都不再执行了，前面的都执行。是这样吧？

【正确回答】：continue 和 break 是指跳出循环，不是跳出 if 语句，如果在没有循环的逻辑中用这两个语句会提示错误的，不过你这种怀疑精神和验证精神是很好的。

至于 exit，我用 PHP 这么久，还没有用过。我在一个 for 循环中试了一下，exit 语句的作用相当于 break，而在实现退出循环时，建议你使用 break。但是在 PHP 中有一个 exit() 函数，用来输出一条消息，并退出当前脚本。

3.8.3 do while 循环和 while 循环的区别



do while 循环和 while 循环的区别

网络课堂：http://bbs.itzcn.com/thread-2230-1-1.html

【问题描述】：请教各个高手给指点一下，do while 循环和 while 循环之间有什么区别？具体在用的时候，我该怎么选择用哪一个呢？

【正确回答】：do while 和 while 循环非常相似，do while 表达式的值是在每次循环结束时检查，而 while 循环是在开始时检查。这样就先执行一次循环体，然后检查循环条件，如果循环条件返回真，则继续执行循环体。

do while 循环语句至少会执行一次。然而 while 循环就不一定了，它首先检查表达式真值，如果值为 true，执行循环体；如果一开始就为 false，则整个循环立即终止。所以，你在用的时候要明白，先执行一次循环体再判断循环条件对程序有没有影响。如果有，那么就选择 while 循环来实现，如果没有，那么使用哪个都可以。

第4章 PHP 类编程

面向对象技术是一种全新设计和构造软件的技术，它使计算机解决问题的方式更符合人类的思维方式，更能直接地描述客观世界，通过增加代码的可重用性、可扩充性和程序自动生成功能来提高编程效率，并且大大减少软件维护的开销，已经被越来越多的软件设计人员所接受。

PHP 5 引入了新的对象模型（Object Model），并且完全重写了 PHP 处理对象的方式，以支持更多的面向对象特性。

本章首先向读者简单阐述面向对象的概念。然后重点对 PHP 中的实现进行介绍，包括创建类、构造函数、类常量、类的方法、PHP 作用域关键字以及继承的实现等。

本章学习要点：

- 理解什么是对象
- 了解面向对象的抽象、封装、继承和多态
- 掌握类的定义和实例化
- 掌握构造函数的使用
- 掌握常量、属性和方法的定义和方法
- 理解 abstract、final 和 protected 关键字的作用域
- 掌握 private、public 和 static 关键字的作用
- 掌握类和构造函数继承的使用

4.1 面向对象简介

面向对象（Object Oriented，OO）是计算机界关心的重点，也是 20 世纪 90 年代软件开发方法的主流。起初，面向对象是专指在程序设计中采用封装、继承、多态等设计方法。

目前，面向对象的概念和应用已超越了程序设计和软件开发，扩展到很宽的范围。面向对象可以分为面向对象的分析（Object Oriented Analysis，OOA）、面向对象的设计（Object Oriented Design，OOD）和面向对象编程（Object Oriented Programming，OOP），并且涉及到数据库系统、交互式界面、应用平台、分布式系统和网络管理结构等领域。

4.1.1 对象的概念

学习面向对象，首先要理解什么是对象。对象是对事物的抽象表示。在面向对象的术

语中，一切皆是对象，一个对象就代表一个具体的功能操作，我们不需要了解这个对象是如何实现某个操作的，只需要知道该对象可以完成哪些操作即可。

一个对象总是具有如下特点。

❑ **万物皆为对象**

例如一本书、一个纸箱、一支笔、一台音箱等，这些都是具体的对象。

❑ **对象都是唯一的**

“世界上没有两个相同的指纹”，在面向对象中也是如此，任何对象都是唯一的。

❑ **对象具有属性和行为**

例如汽车具有品牌、排量、重量、长度、生产日期等属性，还具有发动、行驶、倒车、自动导航等行为。

❑ **对象具有状态**

状态是指某一时刻对象的各个属性的取值。因为对象的属性并不是一直不变的，例如一台汽车的速度和行驶公里数等属性。

❑ **对象都属于某个类别**

每个对象都是某个类别的实例。例如宝马汽车和奥迪汽车都是汽车的实例，都属于汽车类；学生李华和学生陈丽都是学生的实例，都属于学生类。同一个类的所有实例都具有相同的属性，只不过属性的取值不一定相同。例如宝马汽车和奥迪汽车都有外观、车身尺寸、发动机型号等属性，但是这些属性的值不一定相同。

4.1.2 抽象性

“物以类聚，人以群分”是指把众多的事物进行归纳和分类，也是人们在认识客观世界时经常采用的思维方法，而在这里分类所依据的原则是抽象。

抽象（Abstract）就是忽略事物中与当前目标无关的非本质特征，更充分地注意与当前目标有关的本质特征。从而找出事物的共性，并把具有共性的事物划为一类，得到一个抽象的概念。

例如，在设计一个学生管理系统的过程中，考察学生李华这个对象时，就只关心他的学号、班级、成绩等，而忽略他的身高、体重等信息。因此，抽象性是对事物的抽象概括和描述，实现了客观世界向计算机世界的转化。将客观事物抽象成对象及类是比较难的过程，也是面向对象方法的第一步。例如，将学生抽象成对象及类的过程如图 4-1 所示。

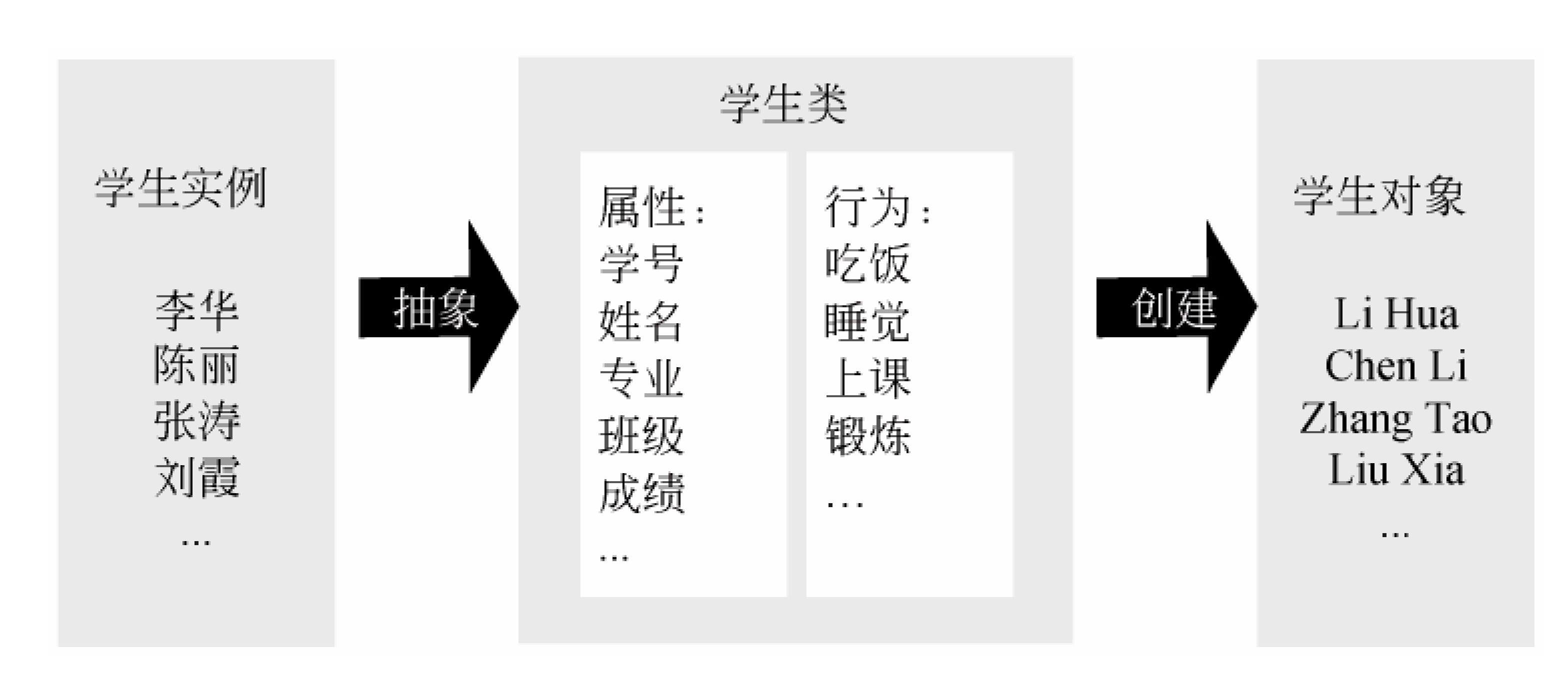


图 4-1　抽象过程示意图

4.1.3 封装性

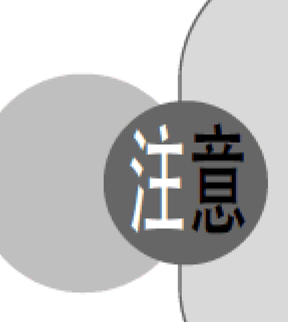
封装（Encapsulation）是指把对象的属性和行为结合成一个独立的单位，并尽可能隐蔽对象的内部细节。例如，图 4-1 中的学生类就实现了封装。

通常来说封装有两个含义：一是把对象的全部属性和行为结合在一起，形成一个不可分割的独立单位，对象的属性值（除了公有的属性值）只能由这个对象的行为来读取和修改；二是尽可能隐蔽对象的内部细节，对外形成一道屏障，与外部的联系只能通过外部接口实现。

封装的信息隐蔽作用反映了事物的相对独立性，可以只关心它对外所提供的接口，即能做什么，而不注意其内部细节，即怎么提供这些服务。例如，对于一台冰箱，不需要知道它具体的实现细节，怎样使用电能控制温度的冷藏与保鲜。只需要知道，怎么打开冰箱，怎么调整温度，怎样存储食品即可。

封装的结果是使对象以外的部分不能随意存取对象的内部属性，从而有效地避免了外部错误对它的影响，大大减小了查错和排错的难度。另一方面，当对象内部进行修改时，由于它只通过少量的外部接口对外提供服务，因此同样减小了内部的修改对外部的影响。

封装机制将对象的使用者与设计者分开，使用者不必知道对象行为实现的细节，只需要用设计者提供的外部接口让对象去做。封装的结果实际上隐蔽了复杂性，并提供了代码重用性，从而降低了软件开发的难度。



如果一味地强调封装，则对象的任何属性都不允许外部直接存取，要增加许多没有其他意义，只负责读或写的行为。这为编程工作增加了负担，增加了运行开销，并且使得程序显得臃肿。为了避免这一点，在语言的具体实现过程中应使对象有不同程度的可见性，进而与客观世界的具体情况相符合。

4.1.4 继承性

客观事物既有共性，也有特性。如果只考虑事物的共性，而不考虑事物的特性，就不能反映出客观世界中事物之间的层次关系，不能完整地、正确地对客观世界进行抽象描述。运用抽象的原则就是舍弃对象的特性，提取其共性，从而得到适合一个对象集的类。

如果在这个类的基础上，再考虑抽象过程中被舍弃的一部分对象的特性，则可形成一个新的类。这个新类具有前一个类的全部特征，是前一个类的子集，形成一种层次结构，即继承结构，如图 4-2 所示。

继承（Inheritance）是一种连接类与类的层次模型。继承性是指特殊类的对象拥有其一般类的属性和行为。继承意味着“自动地拥有”，即特殊类中不必重新定义已在一般类中定义过的属性和行为，而它却自动地、隐含地拥有其一般类的属性与行为。当这个特殊类又被它更下层的特殊类继承时，它继承来的和自己定义的属性和行为又被下一层的特殊类继承下去。因此，继承是传递的，体现了大自然中特殊与一般的关系。

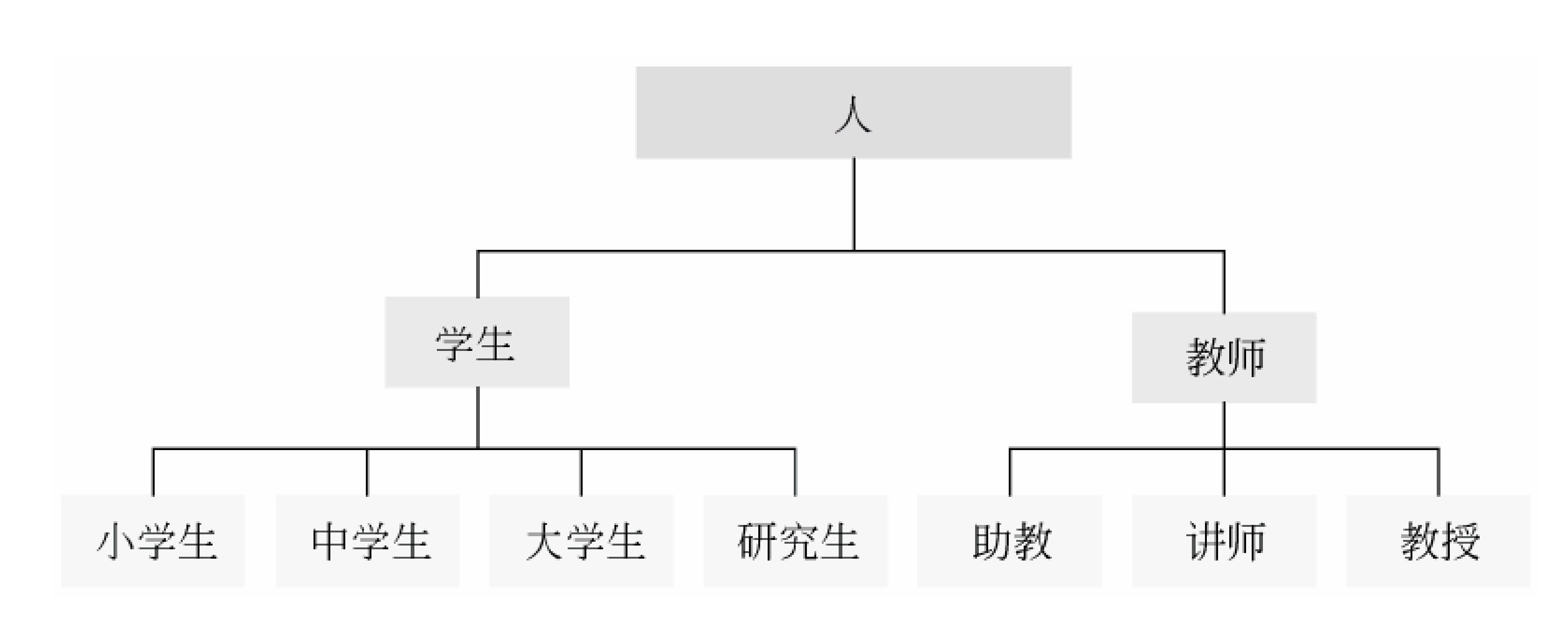


图 4-2　类的继承示意图

在软件开发过程中，继承性实现了软件模块的可重用性、独立性，缩短了开发周期，提高了软件开发的效率，同时使软件易于维护和修改。这是因为要修改或增加某一属性或行为，只需在相应的类中进行改动，而它派生的所有类都自动地、隐含地作了相应的改动。

由此可见，继承是对客观世界的直接反映，通过类的继承，能够实现对问题的深入抽象描述，反映出人类认识问题的发展过程。

4.1.5　多态性

面向对象设计借鉴了客观世界的多态性，体现在不同的对象收到相同的消息时产生多种不同的行为方式。

例如，在一般类“几何图形”中定义了一个行为“绘图”，但并不确定执行时到底画一个什么图形。特殊类“椭圆”和“多边形”都继承了几何图形类的绘图行为，但其功能却不同，一个是要画出一个椭圆，另一个是要画出一个多边形。这样一个绘图的消息发出后，椭圆、多边形等类的对象接收到这个消息后各自执行不同的绘图函数，如图 4-3 所示，这就是多态性的表现。

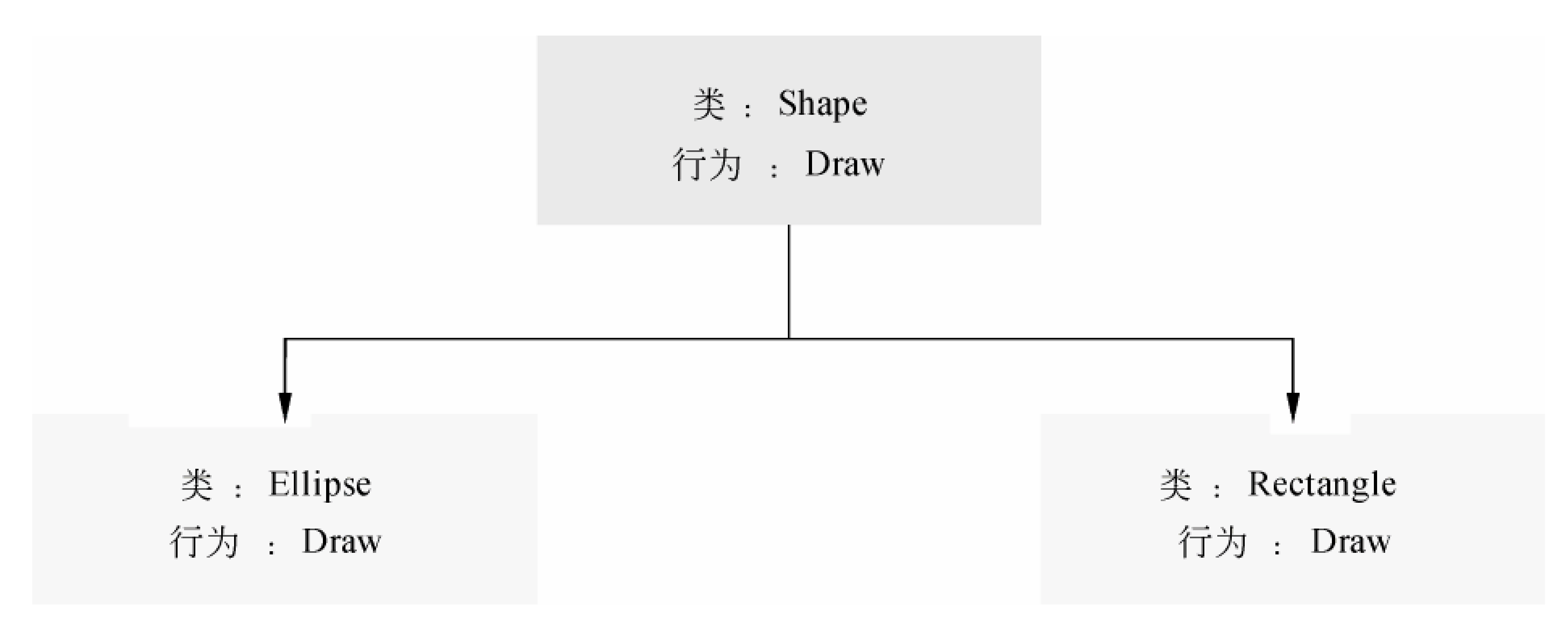
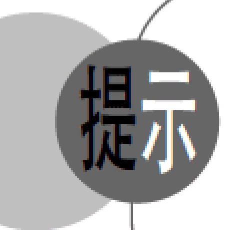


图 4-3　多态性示意图

具体来说，多态性（Polymorphism）是指类中同一函数名对应多个具有相似功能的不同函数，可以使用相同的调用方式来调用这些具有不同功能的同名函数。



继承性和多态性的结合，可以生成一系列虽类似但独一无二的对象。由于继承性，这些对象共享许多相似的特征；由于多态性，针对相同的消息，不同对象可以有独特的表现方式，实现特性化的设计。

4.2 类的基本应用

通过 4.1 节的学习，相信读者一定对面向对象有了简单了解。类（Class）表示要在应用程序中处理的实际事物，是具有相同行为和特点的多个对象的集合。例如，所有的汽车可以看作是一个类，在这个类中又可以划分为不同的类别，如轿车、卡车、货车和公共汽车等。

在面向对象中，将实体的属性和行为组合到一起形成类的定义。本节主要介绍 PHP 中类的一些基本应用，像如何定义类、实例化类（创建类的实体）、类的构造函数和析构函数。

4.2.1 定义类

在 PHP 中定义类的方法非常简单，语法格式如下：

```
class class_name
{
    //这里是类的定义，包括属性和行为等
}
```

其中，class_name 表示要创建的类名称，一般首字母应该大写。在大括号中是类的具体定义，可以在其中编写类的字段、属性和方法等成员。在 4.3 节将会详细介绍这些类成员。

【实践案例 4-1】

根据上面介绍的类定义语法，创建一个名为 Product 的产品类。代码如下所示：

```
<?php
class Product {      //使用 class 关键字声明，Product 表示类名，以下为类的定义
    public $id;
    public $name;
    public $price;
    function toString()
    {
        echo "编号: $this->id , 名称: $this->name , 价格: $this->price";
    }
}                    //Product 类的定义结束
?>
```

在上述创建的类 Product 中有 3 个字段：$id、$name 和$price，以及 1 个函数 toString()。上述代码中的 public 是 PHP 5 中引入的作用域关键字，具体内容将在 4.4 节介绍。

4.2.2 实例化类

对象是类实例化后的结果。在 PHP 中定义好类之后，便可以使用 new 关键字实例化

一个类的对象。其语法格式如下所示：

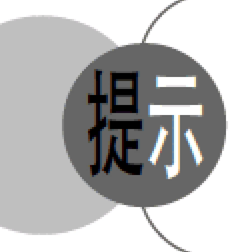
```
$object_name = new class_name();
```

$object_name 表示要创建的对象名称，class_name 是类的名称，跟在类名后面的括号是必需的。

【实践案例 4-2】

根据实例化的语法，创建两个 Product 类的实例，并将对象名称分别命名为$car 和$bus。实现语句如下：

```
$car=new Product();          //创建 Product 类的实例，名为$car
$bus=new Product();          //创建 Product 类的实例，名为$ bus
```



当对象创建完成之后，就可以直接调用类中的方法和字段。一个类可以有多个对象。

4.2.3 构造函数

构造函数在创建类的实例化对象时自动执行，在这里可以对成员进行初始化，或者执行一些特殊操作。PHP 中构造函数的名称被统一命名为__construct()（__是两个下划线）。也就是说，如果在一个类中声明一个命名为__construct()的函数，那么该函数将被当成是一个构造函数，并且在建立对象实例时被执行。与其他语言不同，PHP 中的每个类最多只允许有一个构造函数。

创建构造函数的语法格式如下所示：

```
function  construct([arg1, arg2 , ... , argn]){
    // 构造函数体
}
```

就像其他任何函数一样，构造函数可以有参数或者默认值。

【实践案例 4-3】

在 Product 类中添加一个构造函数用于初始化成员变量的值。代码如下：

```
<?php
class Product {                                    //Product 类
    private $id;                                   //私有成员
    private $name;
    private $price;
    function __construct($id,$name,$price)         //构造函数
    {
        $this->id=$id;
        $this->name=$name;
        $this->price=$price;
```

```
        }
        function toString()
        {
            echo "编号: $this->id , 名称: $this->name , 价格: $this->price \n";
        }
    }
    ?>
```


上述代码的 Product 类由于包含了 3 个私有成员，它们不能在外部访问。所以，创建了一个构造函数来进行初始化，然后可以调用 toString()函数进行输出。测试代码如下所示：

```
$water=new Product("W1-101", "纯净水", 3.5);
                                //通过构造函数传值创建 Product 类的实例
$water->toString();
$cup=new Product("H5-64", "卡通杯", 9.8);
$cup->toString();
```

执行后的输出如下：

```
编号: W1-101 , 名称: 纯净水 , 价格: 3.5
编号: H5-64 , 名称: 卡通杯 , 价格: 9.8
```

4.2.4 析构函数

同构造函数一样，析构函数也有一个统一的命名，即__destruct()（__是双下划线）。析构函数允许在使用一个对象之后执行任意代码来清除内存。默认的仅仅释放对象属性所占用的内存并删除对象相关的资源。

【实践案例 4-4】

下面通过一个例子讲解构造函数和析构函数的使用。在该例子中，计算从类中实例化对象的个数。Counter 类在构造函数中增值，在析构函数中减值。实例代码如下所示：

```
<?php
  class Counter
  {
     private static $count = 0;
     function __construct()          //构造函数
     {
        self::$count++;              //实例化时自动执行
     }
     function __destruct()           //析构函数
     {
        self::$count--;              //释放时自动执行
     }
     function getCount()             //创建一个方法
     {
```

```
        return self::$count;
    }
}
$num1 = new Counter();                  //建立第一个实例
echo $num1 ->getCount()."<br/>";        //输出 1
$num2 = new Counter();                  //建立第二个实例
echo $num2 ->getCount()."<br/>";        //输出 2
$num2 = NULL;                           //销毁实例二
echo $num1 ->getCount();                //输出 1
?>
```

执行后输出结果如下：

```
1
2
1
```



与构造函数不同的是，析构函数__destruct()不接受任何参数。

4.3　类的成员

在 4.2 节学习了如何创建一个类、实例化类、构造函数以及析构函数的使用。本节将详细介绍类成员的使用，即定义在类大括号中的内容。PHP 中的类成员主要可以分为：常量、字段、属性和方法。

4.3.1　常量

在 PHP 的类中使用 const 关键字定义常量。这种方式定义的常量名称前面不需要加“$”修饰符，而且常量名称一般都大写。定义语法格式如下所示：

```
const NAME = "VALUE";
```

这里的 NAME 表示常量的名称，VALUE 表示常量的值。这里的 VALUE 不可以是一个变量、类的成员、数学表达式或函数调用的结果，必须是一个常量表达式。

类常量的访问方式与其他普通变量不同，不能使用对象的实例来访问。如果是在类中访问常量应该使用“self::常量名”语法，在外部访问类中的变量则使用“类名::常量名”语法。

【实践案例 4-5】

下面创建一个案例，使用面向对象的方式来实现计算圆的面积。圆的面积计算公式为：

```
半径*半径*圆周率
```

圆周率又是一个固定的值，因此可以很容易实现。

首先，创建一个表示圆的类 Circle，然后在类中通过常量指定 PI 的值，再创建一个方法来计算结果。具体实现代码如下所示：

```
<?php
class Circle                                 //创建一个 Circle 类表示圆
{
    const PI=3.1415926;                      //常量 PI 表示圆周率
    public function Area($r)
    {
        return self::PI * $r * $r;           //返回面积
    }
}
$s = new Circle();
echo "PI 的值为: ".Circle::PI."\n";          //访问 Circle 类中的常量 PI
echo "所求圆面积是: ".$s->Area(10);
?>
```

如上述代码所示，Circle 类中使用 self::PI 来使用常量 PI，而在类外部必须使用 Circle::PI 来使用常量 PI。执行后输出结果如下：

```
PI 的值为: 3.1415926
所求圆面积是: 314.15926
```

4.3.2　字段

字段用来描述类具有哪些属性，也就是对象所具有的属性，用来表示实体的某一种状态。通常都是在类的开始位置声明字段，并为字段赋初值。

在 PHP 中创建一个字段和创建一个变量的方法基本相同，只不过创建的位置不同。因此，在弱类型的 PHP 中字段也可以不声明而直接创建并赋值，不过不推荐这样使用。

【实践案例 4-6】

创建一个表示学生的 Student 类，然后添加 5 个字段分别学生的姓名、性别、年龄、身高和体重。Student 类的定义如下所示：

```
<?php
class Student{
    public $name;                        //姓名
    var $sex=true;                       //性别
    public $age=22;                      //年龄
    var $height;                         //身高
    var $weight;                         //体重
}
?>
```

上述代码在$name 和$age 字段前面都使用了修饰符指定字段的作用范围。并且在类定

义时$sex 字段初始值为 true，$age 的初始值为 22。

要引用类中的一个字段应该使用“->”操作符。字段只属于某一个对象，所以使用字段时还需要指明该字段是属于哪个对象。使用字段的完整语法格式如下：

```
$object_name->field;    //$object_name 是对象实例名称，field 是要引用字段名称
```

【实践案例 4-7】

根据上面学生类 Student 的定义，创建一个名为$xia 的对象，并输出学生的性别和年龄信息。实现代码如下所示：

```
<?php
class Student{
    public $name;                 //姓名
    var $sex=true;                //性别
    public $age=22;               //年龄
    var $height;                  //身高
    var $weight;                  //体重
}
$xia=new Student();               //实例化$xia 对象
echo "大家好，新手报道。\n";
$sex=$xia->sex?"女":"男";         //获取$sex 字段的值，并转换为字符串“男”或“女”
echo "性别：$sex\n";              //输出性别
echo "年龄：$xia->age";           //输出年龄
echo "\n 结束";
?>
```

由于在 Student 类中将性别$sex 字段定义为布尔型，所以在使用$xia->sex 获取该字段的值之后进行了转换，并将转换后的字符串保存在$sex 变量中。对于$age 字段的年龄信息则可以直接输出，运行后的结果如下所示：

```
大家好，新手报道。
性别：女
年龄：22
结束
```

【实践案例 4-8】

创建一个 Book 类来保存图书信息，包括编号、书名、价格、作者和出版社。然后，创建一个该类的对象进行测试并输出结果。

要实现这个功能有很多种方法，这里使用刚学过的字段来实现，具体代码如下所示：

```
<?php
class Book {
    private $id;                    //私有字段，表示编号
    var $name="未命名";             //书名字段
    public $price=0;                //价格字段
    var $author="未命名";           //作者字段
```

```
    public $pub="未命名";                    //出版社字段
    function  construct($bookid)
    {
        $this->id=$bookid;
    }
    function toAll()                         //输出字段信息
    {
        printf("< %s >这本书很不错哦，推荐你也看看。",$this->name);
        echo "具体信息如下：\n";
        echo "书号：$this->id \n";
        echo "作者：$this->author\n";
        echo "价格：$this->price \n";
        echo "出版社：$this->pub \n";
    }
}
$php=new Book("11640020");                   //创建 Book 类的对象$php
$php->name="PHP 编程基础与实践教程";           //对$name 字段进行赋值
$php->price=39.8;                            //对$price 字段进行赋值
$php->author="祝红涛";                        //对$author 字段进行赋值
$php->pub="清华大学出版社";                    //对$pub 字段进行赋值
$php->toAll();                               //输出信息
?>
```

如上述代码所示，在 Book 类中定义了 5 个字段，还包含一个构造函数以及一个用于输出信息的 toAll()函数。在 toAll()函数中为了引用本类中的字段，需要在前面使用“$this->”前缀。执行结果如下所示：

```
< PHP 编程基础与实践教程 >这本书很不错哦，推荐你也看看。具体信息如下：
书号：11640020
作者：祝红涛
价格：39.8
出版社：清华大学出版社
```

4.3.3 属性

由于 PHP 5 中没有提供属性处理功能，因此需要自己实现这一功能。实现方式主要有两种，第 1 种是通过在类中重载__set()方法和__get()方法来实现对属性的处理。第 2 种则是使用类似 setName()和 getName()的公共方法来获取和设置属性的值。

1. __set()方法

__set()方法用于为隐藏的字段赋值，还可以在为类字段赋值之前添加一些验证数据的代码。语法格式如下所示：

```
boolean __set([String property_name,][mixed value_to_assign])
```

这里包括两个参数，分别表示要设置的属性名和相应属性值。如果执行成功将返回 true，否则将返回 false。

2. __get()方法

__get()方法用于获取类变量的值，它的语法格式如下所示：

```
boolean __get([string property_name])
```

可以看到，该方法只有一个参数，表示要获取的变量名称。如果方法执行成功将返回 true，否则将返回 false。

【实践案例 4-9】

例如，下面创建一个示例说明了如何使用__set()方法和__get()方法来对属性进行操作。实现代码如下所示：

```
<?php
class Rectangle {
    private $Width;                          //私有属性$ Width
    public $Height;                          //公有属性$ Height
    //  set()方法设置属性
    function __set($property_name, $value) {
        echo "自动调用__set()方法为属性 $property_name 赋值\n";
        $this->$property name = $value;
    }
    //__get()方法获取属性
    function  get($property name) {
        echo "自动调用  get()方法获取属性 $property name 的值\n";
        return isset($this->$property name) ? $this->$property name : null;
    }

}
$rect=new Rectangle();
$rect->Width=23;
//直接为私有属性赋值的操作，此时会自动调用__set()方法进行赋值
$rect->Height =40;
//直接获取私有属性的值，此时会自动调用  get()方法，返回属性的值
echo "矩形的高为: ".$rect->Height.", 宽为: ".$rect->Width;
?>
```

在上述代码中，创建了一个类 Rectangle，其中包含了私有属性$Width 和公有属性$Height。另外，还编写了__set()方法和__get()方法重载代码。执行后输出结果如下所示：

```
自动调用__set()方法为属性 Width 赋值
自动调用__get()方法获取属性 Width 的值
矩形的高为: 40, 宽为: 23
```



__set()和__get()前面是两个下划线，而不是一个。如果格式不正确，方法将不能执行。

3. 自定义属性获取和设置方法

使用__set()方法和__get()方法的缺点是无法处理大型且比较复杂的对象属性。因此，PHP 提供了一种由开发人员自定义属性的获取和设置方法。

【实践案例 4-10】

仍然以上个实践案例的 Rectangle 类为例，下面使用这种方式演示如何为私有属性读取和设置值。代码如下所示：

```
<?php
class Rectangle{
    private $Width=0;                              //私有属性$Width
    public $Height=0;                              //公有属性$Height
    function getWidth()                            //获取私有属性$Width 的值
    {
        return $this->Width;
    }
    function setWidth($value)                      //为私有属性$Width 赋值
    {
        $this->Width=$value;
    }
}
$rect=new Rectangle();
echo "矩形的高为: ".$rect->Height.", 宽为: ".$rect->getWidth();
$rect->Height=20;
$rect->setWidth(50);
echo "\n 矩形的新高为: ".$rect->Height.", 新宽为: ".$rect->getWidth();
?>
```

上述代码创建了一个 Rectangle 类，声明了一个私有属性$Width，并为该属性创建了自定义的获取和设置的方法，分别是 getWidth()和 setWidth()。

运行后的输出结果如下所示：

```
矩形的高为: 0, 宽为: 0
矩形的新高为: 20, 新宽为: 50
```

4.3.4 方法

在 PHP 中，类的方法（Method）和程序中的函数比较相似，只不过方法是用来定义类的行为。类中的方法可以完成指定的功能，并具有返回值；同样也可以接受一个输入的参数，并对该数值做出校验再返回结果。

在类中创建方法的语法格式和函数的创建相同，只不过类中的方法都必须使用一些关键字进行定义。声明方法的语法格式为：

```
scope function method name(){
    // 方法体
}
```

在上述代码中，scope 表示方法的作用域范围，可选值有 public、protected 和 private，如果没有设置这些关键字，默认为 public。method_name 表示方法的名称，大括号中表示该方法的执行语句。

【实践案例 4-11】

对于一个计算器来说，执行计算功能应该是最基本的。因此，可以将计算作为计算器的行为，而在面向对象中使用方法来表示对象的行为。

下面就创建一个用于执行计算的计算器类，具体代码如下所示：

```
<?php
class Calculator {                              //定义计算器类
    function clac($number1,$number2,$op)        //定义执行计算的方法
    {
        if($this->check($op))                   //判断操作符是否有效
        {
            switch($op)                         //执行相应的计算
            {
                case "+":
                    $result=$number1+$number2;
                    break;
                case "-":
                    $result=$number1-$number2;
                    break;
                case "*":
                    $result=$number1-$number2;
                    break;
                case "/":
                    $result=$number1/$number2;
                    break;
            }
            echo("操作数 1: $number1 \n");
            echo("操作数 2: $number2 \n");
            echo("操作符: $op \n");
            printf("%d %s %d=%.2f \n",$number1,$op,$number2,$result);
        }
        else{
            echo "出错: 操作符必须是'+,-,*,/'之一。\n";
        }
    }
```

```
    private function check($op)                    //定义操作符检测方法
    {
        return (($op=="+")||($op=="-")||($op=="*")||($op=="/"));
    }
}
?>
```

在 Calculator 类中定义了两个方法，clac()方法前面没有指定关键字，默认使用 public 进行修饰；check()方法使用 private 关键字进行修饰，并返回一个布尔型。其中 clac()方法用于执行计算并输出结果，check()方法用于检查计算时的操作符是否有效。

创建一个方法之后，就可以通过“对象名->方法名()”格式调用类的公共方法。Calculator 类的测试代码如下：

```
$c=new Calculator();
$c->clac(33, 2, "+");                //调用 clac()方法
$c->clac(33, 2, "%");
```

注意，check()方法是私有方法不能直接调用。执行后输出结果如下：

```
操作数 1: 33
操作数 2: 2
操作符: +
33 + 2=35.00
出错: 操作符必须是'+,-,*,/'之一。
```

4.4 作用域关键字

在前面使用的 public 和 private 就是作用域关键字，它们用来修饰类中的方法和字段，每一个都对应一种作用域范围。

PHP 中的作用域关键字有 6 个，分别是：abstract、final、private、protected、public 和 static，下面详细介绍它们。

4.4.1 abstract 关键字

PHP 5 支持抽象类和抽象方法。抽象类不能直接被实例化，必须先继承该抽象类，然后再实例化子类。任何一个类，如果它里面至少有一个方法是被声明为抽象的，那么这个类就必须被声明为抽象的。如果类方法被声明为抽象的，那么其中就不能包括具体的功能实现。

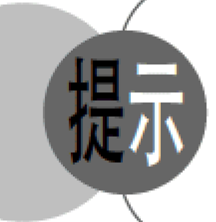
继承一个抽象类的时候，子类必须实现抽象类中的所有抽象方法；另外，这些方法的可见性必须和抽象类中一样。如果抽象类中某个抽象方法被声明为 protected，那么子类中实现的方法就应该声明为 protected 或者 public，而不能定义为 private。

abstract 关键字用来定义抽象方法和类。如下是抽象方法的语法格式：

```
abstract function method_name();
```

使用 abstract 声明一个类 Example，那么该类中的方法都是抽象方法。例如，下面的代码使用 abstract 声明了一个抽象类和三个抽象方法。

```
<?php
abstract class AbstractClass                    //抽象类
{
    abstract function Method1();                //抽象方法
    abstract function Method2();                //抽象方法
    abstract function Method();                 //抽象方法
}
?>
```



抽象类不能够实例化，它的作用类似于接口，就是产生子类的同时给子类一些特定的属性和方法。

4.4.2 final 关键字

final 是 PHP 5 新增的作用域关键字，可以用在类的前面或者方法的前面。当在一个方法的前面加上 final 关键字，表示该方法不可以被重写，即在该类的子类中只允许调用，不允许重新设置该方法的功能。如果一个类声明了 final，那该类不能被继承。

下面的代码演示了如何使用 final 关键字声明方法，以及访问它定义的方法。

```
<?php
class Fish
{
    final function say()        //使用 final 关键字
    {
        echo "我是小龙鱼";
    }
}
Fish::say();                    //直接访问
$f=new Fish();
echo "\nWho are you?\n";
echo $f->say();                 //通过对象访问
?>
```

执行后输出结果如下：

```
我是小龙鱼
Who are you?
我是小龙鱼
```

4.4.3 private 关键字

private 表示私有的，通过 private 关键字修饰的字段和方法只能在类的内部使用，不能通过类的实例化对象调用，也不能通过类的子类调用。如果某些方法是作为另外一些方法的辅助，可以将该方法声明为私有。

private 的使用方式和 public 的使用方式相同，两者只是作用域范围的不同。在 PHP 中访问 private 修饰的成员时必须使用$this 关键字。

【实践案例 4-12】

对于人来说，姓名、性别和出生日期是不能修改的，但是年龄会发生变化。为此，可以创建一个表示人的 Person 类，然后将姓名、性别和出生日期声明为私有的不能修改，只能在类实例化时指定一次。而将年龄作为公共字段。

具体实现代码如下：

```
<?php
class Person {
    private $sex;                          //私有字段，不可直接访问
    private $birthday;                     //私有字段，不可直接访问
    private $name;                         //私有字段，不可直接访问
    var $age=0;                            //公共字段，表示年龄
    function   construct($sex,$birthday,$name)
    {
        $this->sex=$sex;                   //调用私有段
        $this->birthday=$birthday;
        $this->name=$name;
    }
    function toAll()
    {
        echo "我的名字是： $this->name ，今年  $this->age 岁。\n";
        echo "今年  $this->birthday 是我的生日。\n";
        echo "我是一个快乐的 $this->sex 。";
    }
}
//测试 Person 类
$tong=new Person("女孩","10 月 2 日", "桐桐"); //调用构造函数为私有字段赋值
$tong->age=10;                             //调用公共字段
$tong->toAll();
?>
```

执行后输出结果如下：

```
我的名字是：  桐桐  ，今年   10 岁。
今年   10 月 2 日  是我的生日。
我是一个快乐的  女孩  。
```

如果使用实例化对象调用 private 关键字修饰的方法或者字段将导致错误。

4.4.4 protected 关键字

protected 表示受保护的，只能在类本身和子类中使用。使用 protected 关键字修饰字段和方法具有保护作用，通常用来帮助类或子类完成内部计算。如果试图从类外部调用 protected 的成员，将会出现致命性错误。

下面的示例代码演示了如何访问 protected 修饰的方法，实现了查看列车状态当前到站的功能：

```
<?php
class Train
{
    protected $start="北京西";
    protected $end="广州";
    protected function tos()
    {
        echo "本次列车从 $this->start 开往 $this->end ";
    }
    function status()
    {
        $this->tos();
        echo "\n 当前到站：郑州站。";
    }
}
$t=new Train();
$t->status();
?>
```

执行后输出结果如下：

```
本次列车从 北京西 开往 广州
当前到站：郑州站。
```

4.4.5 public 关键字

public 关键字修饰的字段或者方法表示它是公共的，即在 PHP 的任何位置都可以通过对象名来访问该字段和方法。同时 public 也是字段和方法的默认修饰符。

例如，在 School 类中有两个字段，分别是使用 public 声明的$Name 字段和使用默认声明的$CreateDate 字段。该类还包含两个方法，分别是使用 public 声明的 intr()方法和使用默认声明的 word()方法。School 类的具体声明如下所示：

```
<?php
class School
```

```
{
    public $Name="湖南科技大学";
    var $CreateDate="1912 年 9 月 25 日";
    public function intr()
    {
        echo "主要院系： 文学院，教育科学学院，物理与电子学院，信息工程学院，艺术学
        院 \n";
    }
    function words()
    {
        echo "校训：明德，新民，止于至善 \n";
    }
}
?>
```

现在，我们要访问 Student 并调用它的所有成员（2 个字段和 2 个方法）。实现代码如下所示：

```
<?php
echo "通过实例访问 public 方法，输出结果如下：\n";
$s = new School();
echo "$s->Name 创办于 $s->CreateDate \n";
$s->words();
$s->intr();
echo "\n 直接访问 public 方法，输出结果如下：\n";
School::intr();
School::words();
?>
```

注意，使用 public 的字段不能通过类名来直接访问，而方法可以。执行后输出结果如下：

```
通过实例访问 public 方法，输出结果如下：
湖南科技大学 创办于 1912 年 9 月 25 日
校训：明德，新民，止于至善
主要院系： 文学院，教育科学学院，物理与电子学院，信息工程学院，艺术学院

直接访问 public 方法，输出结果如下：
主要院系： 文学院，教育科学学院，物理与电子学院，信息工程学院，艺术学院
校训：明德，新民，止于至善
```

4.4.6 static 关键字

static 关键字用来声明静态成员，可以用在方法或者字段前面。静态成员包括静态方法和静态属性。

使用 static 关键字的静态成员具有如下特性。

- 静态属性不能通过操作符“->”进行访问。
- 由于静态方法可以调用非对象实例，所以$this 关键字不可以在声明为静态的方法中使用。
- 一个类的静态成员会被该类所有的实例化对象共享，任何一个实例化对象对静态成员的修改都会映射到静态成员中。根据这个性质，可以把静态成员看作一个全局变量。
- 静态成员与实例化对象无关，只与类有关。它们用来实现类要封装的功能和数据，但不包括特定对象的功能和数据。
- 静态属性是包含在类中要封装的数据，可以由所有类的实例化对象共享。实际上，除了属于一个固定类并限制访问方式外，类的静态属性非常类似于函数的全局变量。
- 静态方法实现类需要封装的功能，与特定的对象无关。静态方法类似于全局函数。静态方法可以完全访问类的属性，也可以由对象的实例来访问。

下面的示例代码演示了静态变量和静态方法的调用。

```
<?php
class User
{
    static $times=0;                                    //静态变量
    function __construct() {self::$times++;}            //在构造函数修改静态变量的值
    static function log()                               //静态方法
    {                                                   //访问静态变量
        echo "这是第".self::$times."次登录系统\n";
    }
}
echo "初始值: ".User::$times."次\n";                     //使用类名访问静态变量
User::log();                                            //使用类名访问静态方法
$u=new User();
$u->log();
echo "当前值: ".$u::$times."次\n";                       //使用实例访问静态变量
$x=new User();
$y=new User();
$y::log();                                              //使用实例访问静态方法
?>
```

执行后输出结果如下：

```
初始值：0 次
这是第 0 次登录系统
这是第 1 次登录系统
当前值：1 次
这是第 3 次登录系统
```

4.5 对象继承

继承是一个非常重要的面向对象特性，对于功能的设计和抽象非常有用，如果要对类似的对象增加新功能而无需重新编写公用的功能。

PHP 的对象模型也使用了继承，使类和类之间有了相应的层次结构，使类的管理更加清晰，并且提高了代码的可重用性。通常将继承类称为派生类或子类，被继承类称为基类或父类。

4.5.1 继承类

在 PHP 中，类继承通过使用 extends 关键字实现，并且类继承必须是单向继承，也就是说一个类只能有一个基类，但是一个类可以被多个子类继承，继承的语法格式如下所示：

```
class childclass extends parentclass    //childclass 类继承 parentclass 类
{
   //类成员
}
```

上述代码创建了一个继承自 parentclass 父类的子类，子类名称是 childclass，extends 是类的继承关键字。

【实践案例 4-13】

按照面向对象中类封装的思想，可以将“机器”和“手机”分别设计成两个不同的类。而它们又都有一些共同的属性，例如制造商、名称和颜色。手机类只是在机器类的基础上增加了表示自己的属性，例如可以通话。

在实现时如果对这两个类单独设计，不但增加了代码量，还浪费时间和精力。那么该如何处理不同类之间重复代码的问题呢？答案就是使用类的继承特性。

对于本实例，可以首先定义一个表示机器类的 Machine，将它作为父类。代码如下所示：

```
<?php
class Machine                              //父类 Machine
{
    public $color="黑色";                  //颜色
    var $name="基础设备";                  //名称
    var $madeby="广电集团";                //制造商
    function work()
    {
        echo "这是一台  $this->color $this->name , 由 $this->madeby 生产。\n";
    }
}
```

```
?>
```

上述代码为 Machine 类定义了 3 个字段和 1 个方法，在方法中输出了各个字段的值。

接下来，再创建一个表示手机的 Phone 类，为了使它也具有机器类的属性，这里需要让它继承自 Machine 类。代码如下所示：

```
<?php
class Phone extends Machine              //创建子类 Phone
{
    var $model="";
    function call()
    {
        echo "\n 电话正在通话中....\n";
        echo "型号: ".$this->model.", 制造商: ".$this->madeby;
    }
}
?>
```

可以看到，在继承时使用 extends 关键字，后面跟随的是父类名称。在 Phone 类中有一个字段$model 它表示手机的型号，call()方法表示手机的通话功能。

现在 Phone 类继承了 Machine 类，同时它也具有访问 Machine 类中字段和方法的权限。编写代码进行测试，如下所示：

```
<?php
$m=new Machine();                    //创建一台机器
$m->work();                          //输出机器的信息
$p=new Phone();                      //创建一部手机
$p->madeby="长江通信有限公司";        //指定手机制造商
$p->model="IS5800T";                 //指定手机型号
$p->call();                          //调用通话功能
?>
```

执行后输出结果如下：

```
这是一台  黑色 基础设备 , 由 广电集团 生产。

电话正在通话中....
型号: IS5800T, 制造商: 长江通信有限公司
```



子类不但可以拥有父类的成员，如方法和字段，还可以拥有自己本身新增的方法，但是子类不能拥有父类的私有成员。

4.5.2 继承构造函数

子类可以从基类中继承所有公共成员，当然也包括构造函数。在子类中运用基类中的

构造函数与运用其他成员的不同之处在于子类可以显示调用基类中的构造函数，也可以隐式调用基类中的构造函数。

如果父类中有构造函数，并且子类中没有构造函数，那么子类在实例化时自动执行父类构造函数。例如，下面的示例代码演示了这一特性。

```
<?php
class Vehicle{                          //父类
    function __construct(){             //父类构造函数
        echo "我是一辆车\n";
    }
}
class Car extends Vehicle               //子类
{
    var $model;
    function run(){
        echo "$this->model 正在行驶中....";
    }
}
$bmw=new Car();                         //实例化子类
$bmw->model="宝马 730L";
$bmw->run();
?>
```

在代码中，首先创建了一个基类 Vehicle，并且在该类中创建一个构造函数。然后为该类创建了一个子类 Car，在子类中无构造函数。最后实例化子类，查看子类是否调用父类中的构造函数。

执行后输出结果如下：

```
我是一辆车
宝马 730L 正在行驶中....
```

一个类可以调用另外一个类的构造函数，即使这两个类不是基类与子类关系也可以。非继承关系调用构造函数称为显式调用。如果一个子类继承了一个父类，子类只需要使用关键字 parent 就可以直接调用父类构造函数，称为隐式调用。

4.6 项目案例：实现三层架构的用户登录

之前学完 PHP 做了一个用户登录的功能，该功能可以区分普通用户和管理员。普通用户只能查看自己的信息，而管理员可以查看所有用户信息。

当时由于水平和时间有限，没有使用面向对象的方式进行开发。工作后，所有项目都使用了面向对象技术，对这一方式开发的优势也深有体会。所以，最近利用工作之余对这个功能进行了重构，主要是采用三层架构的方式来处理。下面来看看具体实现吧。

【实例分析】

在软件体系架构设计中分层式结构是最常见的，三层架构也是最重要的一种结构。三层分别为：数据访问层（DAL）、业务逻辑层（BLL）和表示层（UI），它体现了软件的“高内聚，低耦合”思想。这是三层架构的定义，限于篇幅，有关三层架构的更多内容就不再介绍。

下面重点分析一下功能，这里主要分为普通用户和管理员两种角色，他们的操作也不一样。

- **普通用户具有编号、姓名、密码、角色、昵称和积分属性。**
- **管理员具有编号、姓名、密码、角色、昵称、积分和 QQ 属性。**
- **普通用户可以登录和查看自己的信息。**
- **管理员可以登录、查看自己的信息、查看所有用户列表。**

根据上面的描述，使用面向对象的抽象原则可以将该功能划分为普通用户类和管理员类。使用封装原则，可以将普通用户和管理员的属性作为类的字段，然后将登录、查看自己和查看列表作为类的方法。另外，为了使类更加容易管理和扩展，这里可以抽象出一个基类表示用户，让普通用户继承该类，管理员又继承普通用户。

这样一来，对象的结构和类的划分就清晰了，也体现了面向对象的抽象、封装和继承。下面开始具体的实现步骤，首先从基础类的创建开始。

（1）新建一个 model.php 文件作为实例的数据模型层。定义一个表示用户的抽象类 User，并添加通用属性，代码如下所示：

```
abstract class User          //用户抽象类
{
    var $Id;                 //编号
    var $Name;               //姓名
    var $Pass;               //密码
    var $Role;               //角色
}
```

（2）从 User 类继承表示普通用户的 M_Member 类，并添加相应属性，代码如下所示：

```
class M_Member extends User
{
    var $Nickname;
    var $Score;
}
```

（3）从 M_Member 类继承表示管理员的 M_Admin 类，并添加相应属性，代码如下所示：

```
class M_Admin extends M_Member
{
    var $QQ;
}
```

（4）在 model.php 文件所在目录新建一个 bll.php 文件作为实例的业务逻辑层。

（5）业务逻辑层主要用于定义对象可以完成哪些操作，并对这些操作进行判断。对于普通用户仅可以登录和查看自己的信息，对应的 B_Member 类代码如下：

```
class B_Member                                        //普通用户业务逻辑类
{
    static function login($member)                    //登录
    {
        if(($member->Name=="zhht")&&($member->Pass=="123456")) return
        true;
        else    return false;
    }
    static function view(){return D_Member::view();}   //查看自己信息
}
```

在上述代码中定义了两个静态方法，login()方法用于登录，view()方法用于显示自己的信息。其中 login()方法包含了判断是否登录成功的逻辑代码，view()方法没有逻辑需要处理直接调用数据访问层返回信息。

（6）创建管理员的业务逻辑类，它比普通用户多了一个查看所有用户的操作。最终代码如下：

```
class B_Admin                                  //管理员业务逻辑类
{
    static function login($admin)              //登录
    {
        if(($admin->Name=="admin")&&($admin->Pass=="admin"))
return true;
        else    return false;
    }
    static function view(){return D_Admin::view();}          //查看自己信息
    static function getList(){return D_Admin::getList();}    //查看所有信息
}
```

（7）由于管理员在本实例中具有最多的操作，所以下面以管理员为例介绍对应数据访问层的实现。新建 dal.php 文件，定义表示管理员的 D_Admin 类。编写逻辑层需要调用的 getList()静态方法，代码如下：

```
class D_Admin                                 //管理员数据访问类
{
    static function getList()                 //获取所有用户列表
    {
        $arr = array(
                array("Id" =>1,"Name" =>"Admin", "Nickname" =>"admin",
                "Pass"=>"123456", "Role" =>"超级管理员","Score"=>100,
                "QQ"=>"26805376"),
```

```
                    array("Id" =>2,"Name" =>"somboy", "Nickname" =>"均焘
                    ","Pass"=>"som114", "Role" =>"副管理员","Score"=>
                    100,"QQ"=>"77552596"),
                    array("Id" =>3,"Name" =>"itzcn", "Nickname" =>"窗内网
                    ","Pass"=>"zz@@#", "Role" =>"栏目管理员","Score"=>
                    100,"QQ"=>"873033910"),
                    array("Id" =>4,"Name" =>"ayhncn", "Nickname" =>"网安
                    ","Pass"=>"cnIje", "Role" =>"新闻管理员","Score"=>
                    100,"QQ"=>"13504141")
                    );
            return $arr;
        }
    }
```

如上述代码所示，静态方法 getList()返回了一个保存有用户列表的数组。在实际开发中需要读取数据库返回信息。

（8）编写查看信息的 view()静态方法，代码如下所示：

```
static function view()
{
    $u=D Admin::getList()[0];
    $str="编号: ".$u["Id"]."<br/>".
            "姓名: ".$u["Name"]."<br/>".
            "昵称: ".$u["Nickname"]."<br/>".
            "密码: ".$u["Pass"]."<br/>".
            "角色: ".$u["Role"]."<br/>".
            "积分: ".$u["Score"]."<br/>".
            "QQ: ".$u["QQ"]."<br/>";
    return $str;
}
```

（9）经过上面的步骤，系统的基础架构就完成了。接下来的工作就是创建系统的表示层即页面，并调用上面的各层。首先新建 index.php 作为系统的登录首页，在这里制作一个表单用于输入用户和密码，以及进行提交。代码如下：

```
<form name="form1" method="post" action="index.php">
    <span class="STYLE1">用户</span><input type="text" name="username" />
    <span class="STYLE1">密码</span><input type="password" name="userpass" />
    <input name="action" type="hidden" id="action" value="login">
    <input type="submit" name="button" id="button" value="登录">
</form>
```

（10）在表单下方添加登录验证的代码。

```
<?php
require 'model.php';
require 'dal.php';
```

```
    require 'bll.php';
    if(isset($_POST["action"]))
    {
        $admin=new M_Admin();                   //实例化一个管理员类
        $admin->Name=$_POST["username"];        //指定用户名
        $admin->Pass=$_POST["userpass"];        //指定密码
        if(B_Admin::login($admin)){             //调用业务层进行登录判断
            $_SESSION["admin"]=$admin;          //保存用户信息
            header("Location: main.html");      //转向主页面
        }
        else{                                   //显示失败信息
            echo "<script>alert('验证失败');</script>";
        }
    }
    ?>
```

如上述代码所示，主要是调用管理员业务逻辑类 B_Admin 的 login()方法进行判断。由于该方法需要一个管理员类实例，所以还需要实例化一个 M_Admin 类并指定用户名和密码。

（11）图 4-4 所示为实例中登录界面的运行效果，图 4-5 所示为登录失败时的提示。



图 4-4　登录界面



图 4-5　登录失败提示

（12）登录成功之后将转到系统的主页面 main.html。在主页面中显示了当前管理员的个人信息，以及所有用户的列表。如下所示为显示个人信息的实现代码：

```
    <?php
    require 'model.php';
    require 'dal.php';
    require 'bll.php';
    if(isset($_SESSION))
    {
```

```
    echo B_Admin::view();                  //调用业务类的 view()方法
}
?>
```

（13）在处理用户列表时需要注意，由于业务类返回的列表是一个数组。因此在这里需要进行保存，然后再遍历输出所需的各项信息。具体代码如下所示：

```
 <?php
require '../model.php';
require '../dal.php';
require '../bll.php';
$all=B_Admin::getList();    //调用业务类的 getList()方法获取所有用户列表
?>
<table width="100%" border="0" cellpadding="0" cellspacing="1"
bgcolor="b5d6e6" onmouseover="changeto()"  onmouseout="changeback()">
        <tr>
          <td width="3%" height="22" background="images/bg.gif"
          bgcolor="#FFFFFF"><div align="center">
            <input type="checkbox" name="checkbox" value="checkbox" />
          </div></td>
          <td width="6%" height="22" background="images/bg.gif"
          bgcolor="#FFFFFF"><div align="center">编号</div></td>
          <td  background="images/bg.gif" bgcolor="#FFFFFF"><div
          align="center">用户名</div></td>
          <td  height="22" background="images/bg.gif" bgcolor=
          "#FFFFFF"><div align="center">密码</div></td>
          <td  background="images/bg.gif" bgcolor="#FFFFFF"><div
          align="center">昵称</div></td>
          <td  background="images/bg.gif" bgcolor="#FFFFFF"><div align=
          "center">角色</div></td>
          <td  background="images/bg.gif" bgcolor="#FFFFFF"><div align=
          "center">积分</div></td>
          <td  height="22" background="images/bg.gif" bgcolor=
          "#FFFFFF"><div align="center">QQ</div></td>
          <td width="15%" height="22" background="images/bg.gif" bgcolor=
          "#FFFFFF" class="STYLE1"><div align="center">基本操作</div></td>
        </tr>
<?php
foreach ($all as $row) {              //遍历用户列表中的每用户
?>
<tr>
    <td height="20" bgcolor="#FFFFFF">
            <input type="checkbox" name="checkbox2" value="checkbox" />
          </td>
          <td height="20" bgcolor="#FFFFFF"> <?php echo $row["Id"]?></td>
          <td height="20" bgcolor="#FFFFFF"><?php echo $row["Name"]?></td>
```

```
            <td height="20" bgcolor="#FFFFFF"><?php echo $row["Pass"]?></td>
            <td bgcolor="#FFFFFF"><?php echo $row["Nickname"]?></td>
            <td bgcolor="#FFFFFF"><?php echo $row["Role"]?></td>
            <td bgcolor="#FFFFFF"><?php echo $row["Score"]?></td>
            <td height="20" bgcolor="#FFFFFF"><?php echo $row["QQ"]?></td>
            <td height="20" bgcolor="#FFFFFF"><img src="images/edt.gif"
            width="16" height="16" />编辑   <img src=
            "images/del.gif" width="16" height="16" />删除</td>
    </tr>
<?php }?>
</table>
```

（14）完成上述步骤之后，实例就制作完成了。最后对系统的页面进行美化。图 4-6 所示为本实例的主页面。



图 4-6　主页面运行效果

4.7　习题

一、填空题

（1）面向对象的应用非常广泛，其中的 OOA 是指__________。

（2）假设有轿车类、卡车类和运输车类，现在创建一个车类可以利用面对象的__________性。

（3）假设定义了一个 Key 类，现在要创建一个名为$k 的该类实例，应该使用代码__________。

（4）如果抽象方法被声明为 protected，那么子类中实现的方法就应该声明为 protected 或者__________。

（5）下面代码的运行结果是__________。

```
class Test{
    function   construct(){
        echo "a";
    }
    function   destruct(){
        echo "b";
    }
}
$t=new Test();
```

（6）在空白处填写代码，使程序运行后输出“itzcn.com”。

```
class Test{
    const WWW="itzcn.com";
}
echo __________;
```

（7）在类中可以使用__________关键字来声明公共成员。

二、选择题

（1）下面不属于面向对象特点的是__________。

A．类是对现实世界对象的抽象

B．具有封装、继承和多态性

C．是针对特定领域的开发模式

D．创建的对象是唯一的

（2）下面关于对象的介绍，不正确的是__________。

A．对象必须是唯一的

B．对象是类的实现

C．对象可以继承

D．对象是过程的实现

（3）下面关键字中不能修饰作用域的是__________。

A．static

B．protected

C．public

D．global

（4）在__________处填写代码，使程序运行后输出“itzcn.com”。

```
class Test{
    var $www="itzcn.com";
    public function tos()
    {
        echo __________;
```

```
    }
}
$t=new Test();
echo $t->tos();
```

A．$this->www

B．self::www

C．self::$www

D．$this::www

（5）下面代码的运行结果是__________。

```
class Class1{
    var $num=0;
    function  construct(){$this->num=1;}
}
class Class2 extends Class1
{
    function write(){ echo $this->num; }
}
$c=new Class2();
$c->write();
```

A．0

B．1

C．null

D．不能执行

三、上机练习

1．制作一个名片夹

使用本章学习的 PHP 面向对象知识制作一个名片夹，具体要求如下。

- 类名为 Contact。
- 包含的字段有编号、姓名、联系电话、邮箱地址和公司名称。
- 包含一个 toString()方法用于输出信息。
- 编写类的测试代码。

2．创建员工类

在一个员工薪酬管理系统中可以将员工工资分为计时、计件、临时、正式、加班和节假日几种，每种方式的发放形式都相同。现在要求读者使用对象的方式来实现这个功能。

3．实现单例模式

本次上机要求读者使用 PHP 实现一个单例模式。

单例模式是指只有一个实例。单例模式确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例，这个类称为单例类。

4.8 实践疑难解答

4.8.1 PHP 类变量的问题



PHP 类变量的问题

网络课堂：http://bbs.itzcn.com/thread-19269-1-1.html

【问题描述】：初学 PHP，对一些概念不理解，求解释。

例如，我怎么在一个方法里使用另一个方法的变量，如：

```
class student{
    static $value=array();
    function a(){
        self: $value[ ]="1";
    }
    function b(){
        print r(self::$value);
    }
}
```

对于上面的代码，我调用 b()方法时并没有输出 a()里面的内容。这是什么原因？如果只要想在 b()里面使用 a()的变量，有没有其他方法。

【解决办法】：首先纠正一点，“调用 a()方法中的变量”这个思路就不对。方法最终只能返回信息，如一字符串或者数组。

方法里面的变量？这个就更有点不能理解了。方法可以接收参数来处理自己本身，一般也只是会初始化一些参数，没有在方法里面定义变量的。原因很简单，方法是用来引用的，引用的时候会传进一些参数供这个方法使用。这个过程是没办法给你定义的变量赋值的。

另外，你要访问的变量$value 是在类中声明的，static 关键字声明它是一个静态变量。访问静态变量应该用 self 关键字。

下面是最终修改后的类和测试代码：

```
class student{
    static $value=array();
    function a(){
        self::$value[]="1";
    }
    function b(){
        print r(self::$value);
    }
```

```
}
$s=new student();
$s->b();
$s->a();
$s->b();
```


4.8.2 关于 PHP 类的私有属性的引用问题



关于 PHP 类的私有属性的引用问题

网络课堂：http://bbs.itzcn.com/thread-19270-1-1.html

【问题描述】：请各位给看看应该怎么解决。下面是代码，有点长：

```
class casher{
    public  function acceptCash($money){    }
}
class CashNormal extends casher{
    public  function acceptCash($money){
        return $money;
    }
}
class CashRebate extends casher{
    private $ rebate=0.1;
    function   constuct($rebate){
        return $this-> rebate=$rebate;
    }
    public function acceptCash($money){
        return $money*$this->_rebate;
    }
}
class CashReturn extends casher{
    private $ offer;
    private $_return;
    function __constuct($offer,$return){
        $this-> offer=$offer;
        $this-> return=$return;
    }
    public function acceptCash($money){
        $result=$money;
        if($money>=$this-> offer){
            $result=$money-floor($money/$this->_offer)*$this->_return;
        }
        //return $result;
        return $this->_offer;                    //此处为什么返回的为 NULL
    }
}
```

```
class CashFactory{
    private static $ cash;
    public static function createCash($type){
        switch($type){
            case "正常收费":
                self::$_cash=new CashNormal();
            break;
            case "满 300 返 100":
                self::$ cash=new CashReturn("300","100");
            break;
            case "打八折":
                self::$ cash=new CashRebate("0.8");
            break;
        }
        return self::$_cash;
    }
}

$cash=CashFactory::createCash("满 300 返 100");
$result=$cash->acceptCash("500");
var_dump($result);
echo $result;
```

这里是创建了一个 casher 基类，并派生了 3 个子类 CashNormal、CashRebate 和 CashReturn。CashFactory 类用于测试。

问题：当方法 acceptCash() 设置为 static 时，在这个方法内引用$this->_offer 和$this->_return 会报不能引用$this 的错误。把 static 去掉后，不再报错，但是引用的值为空，此时还会报 Warning: Division by zero 这个错误，为什么？

【解决办法】：大致看了一下，主要有如下几点问题。

（1）static 不能使用$this 进行访问。

（2）构造函数名称应该是__construct，不是__constuct。

（3）静态方法内不能使用对象的变量，只能调静态变量。

修改之后再重新测试应该没问题了，祝你好运。

第5章 数组处理

使用变量一次只能存储一个值，当需要保存很多相同类型的值时就需要创建很多的变量，这会使程序非常烦琐。为了解决这种问题，可以使用数组来进行处理。

数组（Array）被定义为具有某种共同特性的元素集合，如相似性（会员积分、购物列表、最近 3 天天气等）和类型（所有元素都是字符串或整数）的集合。集合中的所有元素之间由一个特殊的标识符来区分，该标识符通常被称为键（Key）；集合中元素的内容通常被称为值（Value）。

PHP 中的数组与其他语言有所不同。因为 PHP 数组中元素之间可以没有相似性，甚至可以是不同的数据。例如，一个数组可能包含网站的名称、会员总数、手机型号，甚至是学校名称等一些不相关的元素。

本章将详细介绍对数组的各种处理，像测试某个变量是否为数组、遍历数组内容、对数组进行计算和查询，以及数组的排序等。

本章学习要点：

- 掌握数组的创建方法
- 掌握数组的测试和输出方法
- 掌握遍历数组的方法
- 掌握计算数组元素的各种方法
- 掌握元素的增加、删除、定位和提取
- 熟悉数组的截取和合并操作
- 掌握数组排序的各种方法

5.1 创建数组

数组是在程序设计中，为了处理方便，把具有相同类型的若干变量按有序的形式组织起来的一种形式。这些按序排列的同类数据元素的集合称为数组。

在 PHP 中创建数组与创建变量一样都不需要声明。这主要有两种方式，一种是直接赋值，另一种则是使用 array()函数赋值。

5.1.1 使用赋值创建数组

使用赋值方式创建数组是最简单的方式。这种方式实际上就是创建一个数组变量，然

后使用赋值运算符直接给变量赋值，语法形式如下：

```
$arrayName[<key>] = value;
```

其中，arrayName 表示数组变量名，value 表示元素的值，中括号中的 key 表示元素的键。PHP 对数组的键有如下规定。

- **如果没有指定键，则使用默认键，默认键从 0 开始，依次累加。**
- **如果指定了键，则该元素使用指定的键。**
- **如果使用数字或数字型的字符串作为键，则后面的键将以此键为基础开始累加。如果数字是带小数的，按其整数位计算。**

【实践案例 5-1】

使用不带键的赋值方式创建一个数组，代码如下所示：

```
<?php
$waters[]="纯净水";
$waters[]="矿泉水";
$waters[]="苏打水";
$waters[]="营养水";
?>
```

上述代码执行后会创建一个包含 4 个元素的$waters 数组，元素键分别为 0、1、2 和 3。图 5-1 所示为此时$waters 数组的结构。

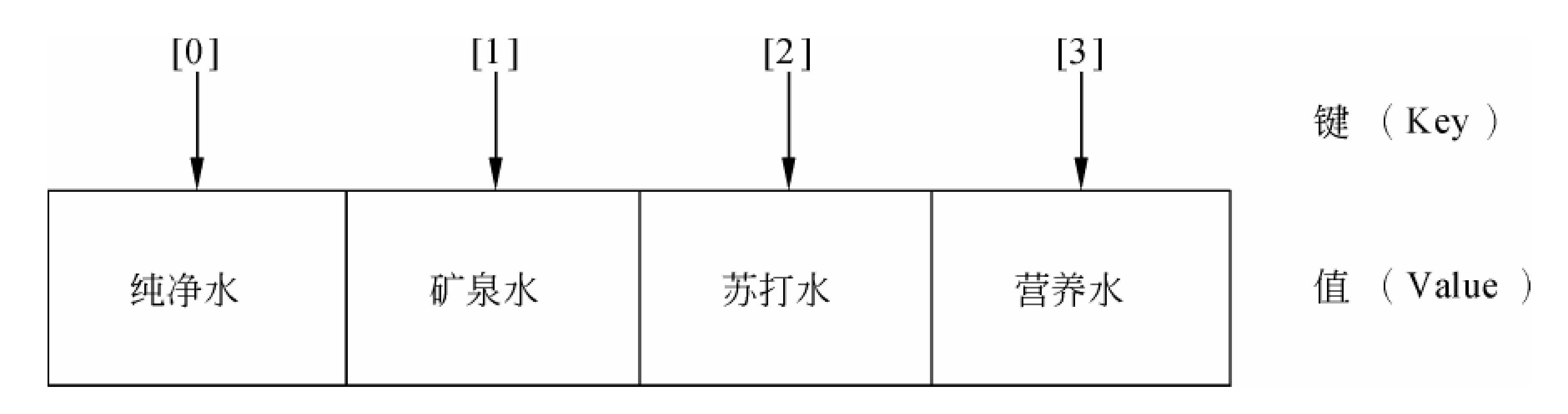


图 5-1　$waters 数组结构示意图



在数组中的键用来唯一标识元素，它有时也被称为索引或者下标。

【实践案例 5-2】

接下来使用指定键的赋值方式创建一个数组，键可以使用数字或者字符串，甚至混合两种的形式。例如下面创建数组的代码：

```
<?php
$teas[6]="龙井";                    //使用数字
$teas["9"]="碧螺春";                //使用数字
$teas["anxi"]="铁观音";             //使用字符串
$teas[]="银针";                     //使用默认方式指定键
$teas[mj]="毛尖";                   //使用字符串
```

```
$teas[y8]="普洱";                        //使用字母+数字的组合
?>
```

上述代码执行后会创建一个名为$teas 的数组，它包含了 6 个元素，图 5-2 所示为此时的数组结构。

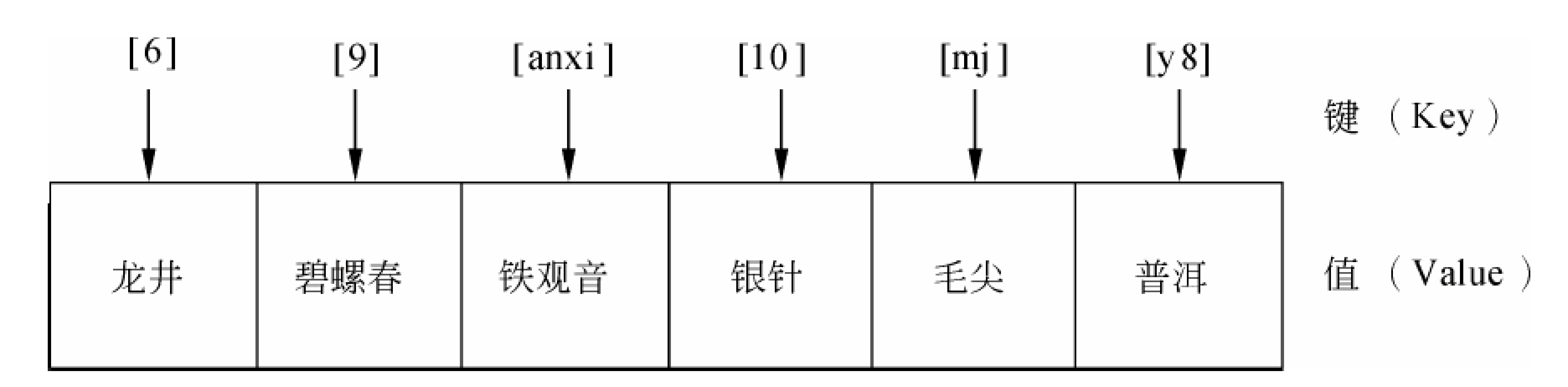
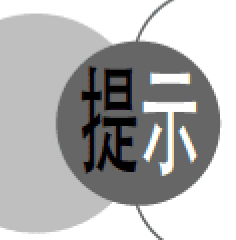


图 5-2 $teas 数组结构示意图



由于指定的键不具有规律性，因此非特殊情况下，推荐使用不指定键的方式进行赋值，这样便于使用循环语句遍历数组。

5.1.2 使用 array()函数创建数组

除了通过赋值方式创建数组外，还可以使用 array()函数创建 PHP 数组。array()函数接收要成为数组的元素作为参数，多个元素之间使用英文逗号（,）分隔。语法形式如下：

```
$arrayName = array(value1 [, value2] [, ...]);
```

可以看到，在使用 array()函数创建数组时变量名后面不要“[]”。

【实践案例 5-3】

使用 array()函数定义两个数组分别用于保存 5 个学生的成绩和姓名。实现代码如下所示：

```
<?php
$scores=array(100,68,79,81,95);                              //成绩数组
$students=array("zhht","som","houxia","ayhncn","lily");      //姓名数组
?>
```

【实践案例 5-4】

使用 array()函数创建数组时，同样也可以指定键，指定格式为“key => value”。例如，使用这种方式重写实践案例 5-2 的$teas 数组，代码如下所示：

```
<?php
    $teas=array(
            6=>"龙井","9"=>"碧螺春","anxi"=>"铁观音","银针",mj=>"毛尖",y8=>"
            普洱"
    );                        //使用 array()创建数组时指定键
?>
```

上述示例中同样混合使用了不指定键与指定键的形式。

5.1.3　创建多维数组

上面介绍的都是一维数组，即看作是一行表格，每个元素是一个单元格。多维数组并不复杂，可以将它看作是由多个一维数组组成的表格，即以多行多列的形式存在。

二维数组就是最简单的多维数组，它指维度为 2 的数组。与普通数组一样，创建多维数组也有两种方式，如下代码演示了如何以赋值方式创建一个二维数组：

```
$one[]="zhht";                //第 1 行 1 列
$one[]=98;                    //第 1 行 2 列
$two[]="somboy";              //第 2 行 1 列
$two[]=84;                    //第 2 行 2 列
$ary[]=$one;
$ary[]=$4two;
```

上述代码执行后将创建 3 个数组，其中$ary 数组包含 2 个元素，第 1 个元素是$one 数组，第 2 个元素是$two。所以$ary 是一个二维数组，其示意如图 5-3 所示。

键：0，0 值：zhht	键：0，1 值：98
键：1，0 值：somboy	键：1，1 值：84

图 5-3　$ary 二维数组示意图

使用 array()函数创建二维数组的方法非常简单，只需使用多个该函数的嵌套形式即可。例如，下面的示例代码：

```
$cities=array(
        array("青岛","济南","日照","威海","烟台"),
        array("长沙","益阳","岳阳"),
        array("安阳","新乡","郑州")
        );
```

执行后$cities 是一个包含 3 个数组元素的二维数组。

5.2　使用数组

创建数组后也就完成了元素的初始化操作，下面介绍两种数组的最简单使用方式。第一种是如何测试一个变量是否为数组，第二种是输出数组的内容。

5.2.1 测试数组

使用 is_array()函数可以测试一个变量是否为数组，语法形式如下：

```
bool is_array(mixed $variable)
```

如果$variable 是数组类型（即使数组中一个元素也没有）则返回 true，否则返回 false。

【实践案例 5-5】

在程序中为了保证能正常执行，首先需要判断某个变量是否为数组，如果不是则先转换再继续。代码如下所示：

```
<?php
$colors="红,黑,白,黄,蓝,绿";                          //定义一个字符串变量
if(!is_array($colors))                                //如果不是数组
{
    echo "\$colors 不是数组，正在转换...";
    $colors=array();                                  //创建一个空数组
    if(is_array($colors)) echo "\n 转换成功，继续执行";//再次判断
}
?>
```

在上述代码中首先定义了一个字符串变量$colors，然后在 if 语句中调用 is_array()测试$colors 是否为数组。如果不是则创建一个空数组并赋给该变量，然后再次判断并输出提示。运行结果如下。

```
$colors 不是数组，正在转换...
转换成功，继续执行
```

5.2.2 输出数组内容

输出一个数组的内容最简单的方法就是使用 print_r()函数，该函数将会按照一定格式显示数组的键和值。语法形式如下：

```
bool print_r ( mixed $arrayname [, bool $return ] )
```

参数$arrayname 是数组的名称，第 2 个参数设置为 true，函数将返回数组内容，而不是输出。

【实践案例 5-6】

假设在$books 数组中保存了若干图书信息，要求输出它的内容，包括数组的键和值。根据所学的知识可以使用 print_r()函数实现，最终代码如下所示：

```
<?php
$books=array("热销图书列表",
        php=>array("PHP 网络大讲堂",34,"PHP 实践教程",46),
```

```
         2=>array(98,40),
         "p"=>"清华大学出版社",
         "08 月 30 日统计",
         "HNZZ110114054"
);
print r ($books);                //输出$books 数组的内容
?>
```

在上述代码中，$books 数组的 php 键又是一个数组，从而形成了嵌套数组。在输出时一块显示出来，结果如下所示：

```
Array
(
    [0] => 热销图书列表
    [php] => Array
        (
            [0] => PHP 网络大讲堂
            [1] => 34
            [2] => PHP 实践教程
            [3] => 46
        )
    [2] => Array
        (
            [0] => 98
            [1] => 40
        )
    [p] => 清华大学出版社
    [3] => 08 月 30 日统计
    [4] => HNZZ110114054
)
```

5.3 遍历数组

在 PHP 中，数组按照键可以分为顺序数组与非顺序数组。所谓顺序数组，是指数组中所有的键是连续的整数；而非顺序数组则是指数组中的键不完全是整数，或者不是连续的整数。下面针对这两种数组介绍遍历方法。

5.3.1 foreach 语句遍历

如果是遍历无顺序的数组，由于数组中的键没有规律可循，所以应该使用 foreach 循环语句，形式如下：

```
foreach($array as $key => $value) {
```

```
    // 循环体
}
```

其中，$array 表示数组，$key 表示键，$value 表示值。

【实践案例 5-7】

假设有一个积分数组使用用户名作为键、积分作为值。现在要求输出每个用户的积分，使用 foreach 遍历的代码如下：

```
<?php
$scores=array("somboy"=>500,xiake=>415,hou=>840,"mary"=>540);  //创建数组
foreach ($scores as $name=>$s) {                               //遍历数组
    echo "$name 的积分是 $s \n";                               //输出用户名和积分
}
?>
```

执行结果如下所示：

```
somboy 的积分是 500
xiake 的积分是 415
hou 的积分是 840
mary 的积分是 540
```

对于顺序数组也可以使用 foreach 循环语句遍历。例如，下面的示例代码：

```
<?php
$keys=array(5=>"php","asp","perl","jsp","python");
                                             //创建数组使用整数有序的键
foreach ($keys as $id=>$keyname) {           //遍历数组
    echo "$keyname 的索引是 $id \n";         //输出键值和键名
}
?>
```

5.3.2　for 语句遍历

遍历顺序数组时，只要知道数组的第一个键值和数组的总长度就可以使用循环语句遍历它。

顺序数组中的第一个键默认值是 0，但是可以被用户设置为任何整数。所以为了确定第一个键到底是多少，可以使用 key()函数。key()函数的语法形式如下：

```
mixed key ( array &$array )
```

它可以返回目标数组$array 中位于当前指针位置的键，第一次调用时自然返回位于第一个位置的键。但是使用时注意 key()函数不会移动指针。对于数组的长度，可以使用 count()函数统计。

【实践案例 5-8】

每个商品上市之后都有它自己的编号。为了方便管理，在进货时会对它分配一个内部

的编号，这些编号是连续的。假设，有一个数组保存了进货后为商品分配的编号和名称，现在要遍历输出它们。

由于数组的键具有连续性，且第 1 个键的值是未知的，因此这里需要借用于 key()函数和 count()函数实现。代码如下所示：

```
<?php
$drinks=array(
    8866=>"和其正","冰红茶","鲜橙多","可乐","雪碧","绿茶"
);                                              //定义商品信息的数组
$min = key($drinks);                            //获取第一个键
$length = count($drinks);                       //获取键的数量
$max = $min + $length - 1;                      //计算键的最大值
for($i = $min; $i <= $max; $i ++){
    echo "编号: $i , 名称: $drinks[$i] \n";    //循环输出
}
?>
```

如上述示例代码，为了不使用默认值 0 作为第一个键，示例中选择使用 8866，然后使用 key()函数获取这个键。循环数组时，需要根据数组的键值访问数组元素，其中最小的键值便是第一个键，而最大的键值则需要根据第一个键与数组的总长度进行计算，公式如下：

```
最小键值 = 第一个键
最大键值 = 第一个键 + 数组长度 - 1
```

执行结果如下所示：

```
编号: 8866 , 名称: 和其正
编号: 8867 , 名称: 冰红茶
编号: 8868 , 名称: 鲜橙多
编号: 8869 , 名称: 可乐
编号: 8870 , 名称: 雪碧
编号: 8871 , 名称: 绿茶
```

5.3.3 each()函数遍历

除了使用 foreach 和 for 语句来遍历数组外，还可以使用 PHP 的数组函数 each()来进行遍历。each()函数语法如下：

```
array each ( array &$array )
```

执行后该函数返回数组的当前“键-值”对，并将指针向下移动一个位置；移动到最后一个元素时返回 false。返回的数组包含 4 个键，键 0 和 key 包含键名，而键 1 和 value 包含相应的数据。

下面演示了使用 each()函数获取数组第 1 个元素的代码：

```
<?php
```

```
    $fruits=array(cm=>"草莓",xc=>"鲜橙",sl=>"石榴",xg=>"西瓜",gj=>"柑橘
    ",lz=>"荔枝");
    $fruit=each($fruits);                        //调用 each()函数获取一个元素
    print_r($fruit);                             //输出 each()函数的返回结果
?>
```


上述代码调用 each()函数从$fruits 数组中获取一个元素。由于该函数的返回值也是一个数组，所以下面使用 print_r()语句输出数组内容，结果如下：

```
Array
(
    [1] => 草莓
    [value] => 草莓
    [0] => cm
    [key] => cm
)
```

从结果中可以看到，在 each()返回的数组中包含 4 个元素，其中键 0 和 key 包含键名，而键 1 和 value 包含相应的数据。

如果再次调用 each()函数将会输出$fruits 数组的第二个元素。根据这个特性再结合 while 语句可以很容易地遍历数组。如下所示为$fruits 数组的遍历代码：

```
$fruits=array(cm=>"草莓",xc=>"鲜橙",sl=>"石榴",xg=>"西瓜",gj=>"柑橘",lz=>"
荔枝");
while($item=each($fruits))
{
    echo "水果名称: $item[value] ,简称:  $item[key] \n";
}
```

执行时每次都从$fruits 数组中获取一个元素保存到$item 中，当读取完成后返回 false 结束循环。执行后的输出结果如下：

```
水果名称: 草莓 ,简称:  cm
水果名称: 鲜橙 ,简称:  xc
水果名称: 石榴 ,简称:  sl
水果名称: 西瓜 ,简称:  xg
水果名称: 柑橘 ,简称:  gj
水果名称: 荔枝 ,简称:  lz
```

each()函数还可以跟 list()函数一块使用实现数组的遍历。例如，下面的代码：

```
while(list($key,$value)=each($fruits))
{
    echo "水果名称: $value ,简称:  $key \n";
}
```

5.3.4 遍历数组函数

除了前面用到的 each()函数以外，PHP 中还提供了很多遍历数组时可以使用的函数，

这些函数如表 5-1 所示。

表 5-1　遍历数组时可以使用的函数

函数	说明
reset()	该函数用来将数组的指针设置回数组的开始位置。如果需要在一个脚本中多次查看或处理一个数组，就经常使用这个函数，另外这个函数还经常在排序结束时使用
current()	该函数返回位于数组当前指针位置的数组值。与 next()、prev()和 end()函数不同，current()不移动指针
end()	该函数将指针移向数组的最后一个位置，并返回最后一个元素
next()	该函数返回紧接着放在当前数组指针的下一个位置的数组值
prev()	该函数返回位于当前指针前一个位置的数组值，如果指针本来就位于数组的第一个位置，则返回 false
array_reverse()	该函数将数组中元素的顺序逆置。如果设置其可选参数为 true，则保持键映射。否则，重新摆放后的各个值将对应于先前该位置上的相应键
array_flip()	该函数将使数组中键及其相应值互换角色

5.4　数组计算

5.3 节学习了遍历数组元素的几种方式，它们可以将元素输出。本节将介绍针对数组元素常用的计算操作，像计算元素总个数、出现的次数等。

5.4.1　计算元素总数

数组在经过初始化和多次操作之后，其中的元素数量可能会发生变化。为了获取数组中元素的数量，可以使用 count()函数。该函数语法形式如下：

```
int count ( mixed $var [, int $mode ] )
```

$mode 是可选参数，用于设置是否进行递归计数，如果设置其值为 1，则进行递归计数；如果不设置该参数，或者设置其值为 0，则不进行递归计数。

【实践案例 5-9】

假设，在一个数组中保存了若干的城市名称，要统计出城市的数量。编写代码并使用 count()函数，如下所示：

```
<?php
$students=array("李雷","张强","刘浩","陈勇",
    "mm"=>array("李晶晶","李金金"),
    b=>array("王珂")
);
$numbers=count($students);              //统计个数，不启用递归
echo "当前共有 $numbers 个学生";
?>
```

执行结果如下所示：

```
当前共有 6 个学生
```

从输出结果中可以看到，在默认使用 count()函数时没有启用递归。此时，如果数组元素是一个数组，那么将会忽略。所以 mm 数组中的元素不计数，输出 6。

在上面代码中添加如下两行代码，重新统计个数，这里启用递归计数。

```
<?php
$result2=count($students,1);                          //统计个数，启用递归
echo "\ncount(\$students,1)的结果为: $result2";
?>
```

执行结果如下所示：

```
count($Array,1)的结果为: 9
```

可以看到，这样输出的会将 mm 数组和 b 数组中的元素也统计在内。所以，最终输出 9。

5.4.2 计算元素出现的频率

在 PHP 中，数组元素是允许重复的，使用 array_count_values()函数可以计算每个元素出现的频率。函数语法形式如下：

```
array array_count_values ( array $input )
```

该函数返回一个包含“键-值”对的数组，其中，键表示$input 数组中元素的值，而值则是该元素在数组中出现的次数。

【实践案例 5-10】

假设在$keys 数组中保存了用户最近搜索的 10 个关键字，现在要统计每个关键字出现的次数。这里统计元素在数组中出现的频率要用到 array_count_values()函数，实现代码如下所示：

```
<?php
//关键字数组
$keys=array(2012,"奥运","电影",2012,"安卓","科技","计算机","安卓","计算机
",2012);
$result=array_count_values ($keys);     //统计$keys 数组中的元素出现频率
while(list($key,$value)=each($result))
{
    echo "关键字[$key]一共搜索  $value 次 \n";
}
?>
```

执行结果如下所示：

```
关键字[2012]一共搜索  3 次
关键字[奥运]一共搜索  1 次
```

```
关键字[电影]一共搜索  1 次
关键字[安卓]一共搜索  2 次
关键字[科技]一共搜索  1 次
关键字[计算机]一共搜索  2 次
```


5.4.3　计算出现的所有元素

前面学习了如何计算元素出现的频率，那如果要去除重复的元素应该怎么办呢？可以使用 array_unique()函数实现数组中元素的唯一性，语法形式如下：

```
array array_unique(array $input_array)
```

该函数返回一个由$input_array 数组中唯一值所组成的数组。

同样以 5.4.2 节的关键字数组为例，要去除重复出现的关键字，实现代码如下所示：

```
<?php
//关键字数组
$keys=array(2012,"奥运","电影",2012,"安卓","科技","计算机","安卓","计算机
",2012);
$result=array_unique($keys);                //去除重复的元素
echo "热门关键字有: ";
foreach ($result as $key) {
    echo " $key";
}
?>
```

执行结果如下所示：

```
热门关键字有: 2012 奥运 电影 安卓 科技 计算机
```

5.5　数组元素操作

在 PHP 中，针对数组元素的操作主要有增加、删除、定位和提取。PHP 为这些操作提供了很多内置函数，下面将详细介绍它们。

5.5.1　增加元素

要向数组中增加一个元素，最简单的操作就是直接赋值，这在创建数组时已经介绍过。例如，下面的 3 行代码都实现了向$color 数组中添加一个元素：

```
$colors[]="红色";              //使用默认键添加元素
$colors['red']="红色";         //使用字符串键添加元素
$colors[2]="红色";             //使用数学键添加元素
```

除了上面的 3 种方式，PHP 还提供了 3 个用于实现增加数组元素的函数。

1. array_push()函数

array_push()函数可以向目标数组的末尾添加一到多个元素，其语法形式如下：

```
int array_push(array $target_array, mixed $variable [, mixed $variable …])
```

该函数返回添加元素后数组的新长度。

【实践案例 5-11】

假设在创建$courses 数组时定义了一学期的主修课程，选修课确定之后也需要添加到该数组中。最终输出本学期的课程数量以及课程名称，实现代码如下：

```
<?php
$courses=array("语文","数学","英语");                   //初始化课程数组
echo "本学期的主修课有".count($courses)."门 \n";         //输出当前元素数量
array_push($courses, "体育","美术","课外实践","自然");    //增加 4 个元素
echo "加上选修课，本学期课共有".count($courses)."门 \n";
echo "分别是：";
foreach ($courses as $c) {
    echo " $c";                                         //输出元素值
}
?>
```

执行结果如下所示：

```
本学期的主修课有 3 门
加上选修课，本学期课共有 7 门
分别是： 语文 数学 英语 体育 美术 课外实践 自然
```

2. array_unshift()函数

array_unshift()函数的功能与 array_push()函数相反，它可以将一个或多个元素添加到数组的开始位置，其语法形式如下：

```
int array_unshift(array target_array, mixed variable [, mixed variable …])
```

该函数同样返回添加元素后数组的新长度。例如，使用 array_unshift()替换实践案例 5-11 中的 array_push()将得到如下输出：

```
本学期的主修课有 3 门
加上选修课，本学期课共有 7 门
分别是： 体育 美术 课外实践 自然 语文 数学 英语
```

下面代码演示 array_unshift()函数向数组中增加元素的使用方法：

```
<?php
$letters=array("A","B","C");
```

```
array_unshift($letters,"K","G","T");        //向数组中添加元素
print_r($letters);                          //输出数组的内容
?>
```

执行结果如下所示：

```
Array
(
    [0] => K
    [1] => G
    [2] => T
    [3] => A
    [4] => B
    [5] => C
)
```

3. array_pad()函数

array_pad()函数可以向目标数组中填充指定元素，直到数组的长度达到指定大小为止，其语法形式如下：

```
array array_pad(array $target_array, integer $length, mixed $pad_value)
```

如果$length 参数为正值，则表示将指定元素填充到数组的右侧；为负时表示将元素填充到数组的左侧。



如果$length 参数的绝对值小于等于数组的当前长度，则不执行操作。

例如，下面的代码使用 array_pad()函数分别向$files 数组的左侧和右侧进行填充：

```
<?php
$letters=array("D","M");
$new_letters = array_pad($letters, 4, "*");
                                       //在 array_pad()函数中使用正值
print_r($new_letters);
$new_letters = array_pad($letters, -4, "*");
                                       //在 array_pad()函数中使用负值
print_r($new_letters);
?>
```

执行结果如下所示：

```
Array
(
    [0] => D
    [1] => M
```

```
    [2] => *
    [3] => *
)
Array
(
    [0] => *
    [1] => *
    [2] => D
    [3] => M
)
```

5.5.2 删除元素

要从数组中删除元素，最常用的是 array_pop()函数和 array_shift()函数。它们分别用于从数组末尾删除一个元素和从数组开头删除一个元素。

1. array_pop()函数

array_pop()函数可以从目标数组中取出最后一个元素，其语法形式如下：

```
mixed array_pop(array $target_array)
```

删除元素后会将$target_array 数组的长度减一，并返回取出的元素。如果$target_array 为空或者不是数组将返回 NULL。

【实践案例 5-12】

例如，下面的代码使用 array_pop()函数从$books 数组的末尾取出一个元素，然后输出该元素和数组的内容：

```
<?php
$books=array("西游记","三国演义","水浒传","红楼梦");
$book= array_pop($books);                //取出$books 数组中的最后一个元素
echo "从\$books 数组中删除了[".$book."]元素";       //输出取出的元素
print r($books);
?>
```

执行结果如下所示：

```
从$books 数组中删除了[红楼梦]元素
Array
(
    [0] => 西游记
    [1] => 三国演义
    [2] => 水浒传
)
```

2. array_shift()函数

array_shift()函数的功能与 array_pop()函数相反，它可以从目标数组中取出第一个元素，

其语法形式如下：

```
mixed array_shift(array $target_array)
```

例如，使用 array_shift()替换实践案例 5-12 中的 array_pop()函数，将看到如下的输出：

```
从$books 数组中删除了[西游记]元素
Array
(
    [0] => 三国演义
    [1] => 水浒传
    [2] => 红楼梦
)
```

在使用 array_shift()函数时要注意，取出第一个元素后数组中数字类型的键将自动进行重新调整，非数字类型的键不受影响。同样，如果参数为空或者不是数组将返回 NULL。

例如，下面的代码演示了使用 array_shift()函数前后带数字键数组的变化：

```
<?php
$cities=array(100=>"安阳",zz=>"郑州","开封",ly=>"洛阳","新乡");
print_r($cities);                                    //输出原始数组
$city= array_shift($cities);                         //删除一个元素
echo "从\$waters 数组中删除了元素: $city \n";
print_r($cities);                                    //输出删除元素后数组
?>
```

执行结果如下所示：

```
Array
(
    [100] => 安阳
    [zz] => 郑州
    [101] => 开封
    [ly] => 洛阳
    [102] => 新乡
)
从$waters 数组中删除了元素: 安阳
Array
(
    [zz] => 郑州
    [0] => 开封
    [ly] => 洛阳
    [1] => 新乡
)
```

5.5.3 定位元素

定位元素是指对数组中的元素进行有效的搜索、筛选和查找，并最终返回符合条件的

元素。PHP 中实现定位元素的函数主要有：array_keys()、in_array()、array_search()、array_values()和 array_keys_exists()。

1. array_keys()函数

array_keys()函数可以返回数组中所有由键所组成的数组，语法形式如下：

```
array array_keys (array $target_array [, mixed $search_value])
```

通过$search_value 可选参数可以指定搜索值，此时将只返回与该值相等的元素。

【实践案例 5-13】

假设有一个保存用户积分的数组$scores，现在要从中获取积分为 100 的元素并组成一个新数组，实现代码如下所示：

```
<?php
$scores = array('murphy' => 92,'lelei' => 90, 'chengj' => 100, 100, 'forever'
=> 89);
$result1 = array_keys($scores);              //返回带键的数组
print_r($result1);                           //输出结果
$result2 = array keys($scores, 100);         //返回带键，且值为 100 的数组
print r($result2);                           //输出结果
?>
```

执行结果如下所示：

```
Array
(
    [0] => murphy
    [1] => lelei
    [2] => chengj
    [3] => 0
    [4] => forever
)
Array
(
    [0] => chengj
    [1] => 0
)
```

第一次使用 array_keys()函数时没有指定要搜索的值，此时返回了$scores 数组中所有元素的键。第二次使用 array_keys()函数时指定搜索值为 100，所以只返回了$score 数组中值为 100 元素的键。

2. in_array()函数

in_array()函数可以在目标数组中查找指定值对应的元素，语法形式如下：

```
bool in_array(mixed $needle, array $haystack [, bool $strict])
```

执行时会先在$haystack 数组中搜索$needle（区分大小写），如果找到则返回 true，否则返回 false。下面的代码演示了 in_array()函数搜索字符串的用法：

```
<?php
$words=array("hello","OK","world","good");
echo "在\$words 数组中查找是否包含 hello: ";     //查找 hello 字符串，区分大小写
var_dump(in_array( "hello",$words));         //返回 true
echo "在\$words 数组中查找是否包含 ok: ";       //查找 ok 字符串，区分大小写
var_dump(in_array( "ok",$words));            //返回 false
?>
```

由于 in_array()函数区分大小写，所以上面第 2 个查询会失败。最终执行结果如下所示：

```
在$words 数组中查找是否包含 hello: bool(true)
在$words 数组中查找是否包含 ok: bool(false)
```

in_array()函数的第 3 个参数$strict 用于设置是否在搜索时考虑数据类型，如果不设置此参数，则默认不考虑数据类型，如果设置其为 1 或 true，则考虑数据类型。下面的代码演示了 in_array()函数搜索时区分数据类型的用法：

```
<?php
$scores=array(100,81,69,71,'81.8',92.5);
echo "查找\$scores 数组中是包否含有 92.5: \n";
if (in_array(92.5, $scores, true)) {             //区分数据类型和大小写
    echo "找到 92.5 了。\n ";
}
echo "\n 查找\$scores 数组中是否包含有 100: \n";
if (in_array('100', $scores, true)) {            //区分数据类型和大小写
    echo "找到 100 了。";                         //此句不会输出，因为类型不一致
}
?>
```

在第 2 次查询中由于设置 in_array()函数的第 3 个参数为 true，此时将对类型进行判断，所以'100'=100 结果为 false，不会输出“找到 100 了。”。

最终执行结果如下所示：

```
查找$scores 数组中是包否含有 92.5:
找到 92.5 了。
查找$scores 数组中是否包含有 100:
```

3. array_search()函数

array_search()函数可以在目标数组中查找指定值并返回它的键，语法形式如下：

```
mixed array_search (mixed $needle, array $haystack [, bool $strict] )
```

如果在$haystack 数组中找到了值$needle，则返回其键，否则返回 false。可选参数$strict

同样用于设置是否在搜索时考虑数据类型。使用时要注意，如果多个键对应相同的值，只返回第一个键。

现在要从保存会员积分的$scores 数组中查找积分为 100 的会员信息，实现代码如下：

```
<?php
$scores = array('murphy' => 92,'lelei' => 90, 'chengj' => 100, 100, 'forever'
=> 89);
$name = array_search(100, $scores);      //查找$scores数组中是否包含100这个值
echo "积分为 100 的会员有: ".$name;
?>
```

在代码定义的$scores 数组中，虽然 100 不只出现一次，但 array_search()函数仅会返回第一个满足的键，即 chengj。执行结果如下所示：

```
积分为 100 的会员有: chengj
```

4. array_values()函数

array_values()函数通常用于为数组重新建立索引，它可以返回数组中的所有值，并自动为返回的数组提供数值索引。该函数的语法形式如下：

```
array array_values(array $target_array)
```

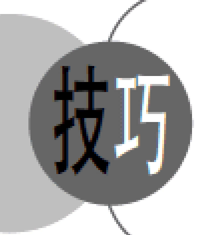
这里同样有一个保存会员积分的数组$scores，但是它使用的键没有规律，有的是字符串，有的是整数。现在想输出它包含的元素，可以使用 array_values()函数将元素的值放到一个新数组中再遍历。实现代码如下：

```
<?php
$scores = array('murphy' => 92,'lelei' => 90, 'chengj' => 100, 151=> 100,
'forever' => 89);
print r($scores);
$new scores= array values($scores);      //将所有值保存到$new scores数组
print_r($new_scores);
?>
```

执行结果如下所示：

```
Array
(
    [murphy] => 92
    [lelei] => 90
    [chengj] => 100
    [151] => 100
    [forever] => 89
)
Array
(
```

```
    [0] => 92
    [1] => 90
    [2] => 100
    [3] => 100
    [4] => 89
)
```



使用 array_values()函数返回的新数组有利于对数组进行操作，例如遍历和排序。

5. array_keys_exists()函数

array_keys_exists()函数可以判断某个键是否属于指定数组，其语法形式如下：

```
bool array_key_exists(mixed $key, array $target_array)
```

如果 key 属于 target_array 数组则返回 true，否则返回 false。

同样是保存会员积分的数组$scores，它使用会员名作键、积分作值。现在要从中查找是否包含有会员名 admin 的积分，如果有就输出。实现代码如下所示：

```
<?php
$scores = array('murphy' => 92,'lelei' => 90, 'chengj' => 100,'admin'=> 100,
'forever' => 89);
if(array_key_exists('admin', $scores)){          //判断$score 数组中是否
包含有 admin 键
    echo "会员 admin 的积分是: ".$scores[admin];
}else{
    echo "找不到会员 admin";
}
?>
```

执行结果如下所示：

```
会员 admin 的积分是: 100
```

5.5.4 提取元素

PHP 提供了 4 个函数用于从多个数组中提取其他数组中没有的元素和提取共有的元素。它们分别是 array_diff()、array_diff_assoc()、array_intersect()和 array_intersect_assoc()。

1. array_diff()函数

array_diff()函数用于从某个数组中提取其他指定数组中不包含的元素，其语法形式如下：

```
array array_diff(array $input_array1, array $input_array2 [, ..., array
$input_arrayN])
```

该函数以$input_array1 为基础数组，返回$input_array1 中不被其他数组所包含的元素。

【实践案例 5-14】

假设有两个数组分别存储了两天内最活跃的 5 个会员名称。现在要从中找出在第 1 天出现，但没有出现在第 2 天的会员。

这就需要使用 array_diff()函数从数组中提取不重复的元素，实现代码如下：

```
<?php
$users1=array("mary","xiaoqiang","laoshu","java163","murphy");
                                                //会员数组 1
$users2=array("laoshu","notok","murphy","isbest","java163");
                                                //会员数组 2
$result = array_diff($users1, $users2);         //提取不相同的会员
print_r($result);
?>
```

执行结果如下所示：

```
Array
(
    [0] => mary
    [1] => xiaoqiang
)
```

2. array_diff_assoc()函数

array_diff_assoc()函数的作用与 array_diff()函数相同，不同之处在于它在比较时是使用“键-值”对，而不只是值，其语法形式如下：

```
array array_diff(array $input_array1, array $input_array2 [, ..., array
$input_arrayN])
```

对实践案例 5-14 的两个数组进行修改，使用会员名作键、编号作为值。要从中找出在第 1 天出现，但没有出现在第 2 天的会员。下面的代码演示了使用 array_diff_assoc()函数实现的过程：

```
<?php
$users1=array("mary"=>30,"xiaoqiang"=>27,"laoshu"=>39,"java163"=>21,"mu
rphy"=>25);                                     //会员数组 1
$users2=array("laoshu"=>39,"notok"=>4,"murphy"=>25,"isbest"=>52,"java16
3"=>21);                                        //会员数组 2
$result = array_diff_assoc($users1, $users2);   //提取不同的元素
print_r($result);                               //输出结果
?>
```

执行结果如下所示：

```
Array
(
    [mary] => 30
    [xiaoqiang] => 27
)
```

3. array_intersect()函数

array_intersect()函数用于提取多个数组中都包含的元素，并将这些元素合并成一个数组，其语法形式如下：

```
array array_intersect(array $input_array1, array $input_array2 [, ..., array
$input_arrayN])
```

以实践案例 5-14 的数组为基础，现在要找出同时出现在这两天的会员名称。使用 array_intersect()函数的实现代码如下：

```
<?php
$users1=array("mary","xiaoqiang","laoshu","java163","murphy");//会员数组 1
$users2=array("laoshu","notok","murphy","isbest","java163");//会员数组 2
$result = array_intersect($users1, $users2);   //提取相同的会员
print_r($result);
?>
```

执行结果如下所示：

```
Array
(
    [2] => laoshu
    [3] => java163
    [4] => murphy
)
```

4. array_intersect_assoc()函数

array_intersect_assoc()函数与 array_intersect()函数作用相同，不同之处在于它在比较元素是否相同时比较的不仅仅是值，而是键-值对。其语法形式如下：

```
array array_intersect_assoc(array $input_array1, array $input_array2 [, ...,
array $input_arrayN])
```

对实践案例 5-14 的两个数组进行修改，使用会员名作键、编号作为值。现在要找出同时出现在这两天的会员名称。使用 array_intersect_assoc()函数的实现代码如下：

```
<?php
$users1=array("mary"=>30,"xiaoqiang"=>27,"laoshu"=>39,"java163"=>21,"mu
```

```
rphy"=>25); //会员数组 1
$users2=array("laoshu"=>39,"notok"=>4,"murphy"=>25,"isbest"=>52,"java16
3"=>21);    //会员数组 2
$result = array_intersect_assoc() ($users1, $users2);  //提取同相同的会员
print r($result);                                     //输出结果
?>
```


执行结果如下所示：

```
Array
(
    [laoshu] => 39
    [java163] => 21
    [murphy] => 25
)
```

5.6 数组操作

在学习了如何对数组元素进行增加、删除、定位以及提取之后，本节将介绍常用的两种数组操作：合并和截取。同其他操作一样，PHP 提供了很多内置函数方便开发人员的调用，在接下来的内容中将对它们进行介绍。

5.6.1 截取数组

PHP 中不仅可以从一个数组中截取掉一部分元素，还可以使用其他元素填充被截取掉的部分。

1. array_slice()函数

array_slice()函数可以从目标数组中截取一部分元素，其语法形式如下：

```
array array slice ( array $array , int $offset [, int $length [, bool
$preserve_keys ]] )
```

各个参数的含义如下。

- $offset 用于指定起始位置。如果$offset 为正，则将从$input_array 数组中距离数组开头的$offset 位置开始截取。如果$offset 为负，则将从$input_array 数组中距离数组末尾的$offset 位置开始截取。
- $length 用于指定截取长度。如果$length 为正，则会在距数组开头的$offset+$length 位置结束截取。如果$length 为负，则在距数组开头的 count($input_array) - $length 位置结束截取。如果省略$length，则将从$offset 开始一直截取到$input_array 数组的末尾。
- $preserve_keys 用于设置是否重置数组的键。

截取时包含起始位置的元素，但不包含结束位置的元素。另外，数组中的元素位置从0开始计算。下面的代码演示了如何使用 array_slice()函数对$cities 数组进行各种截取操作：

```
<?php
$cities=array("北京","上海","郑州","南京","杭州","长沙","青岛");
$result = array slice($cities, 2);
echo "从索引为 2 的元素开始截取产生的数组: \n";
print_r($result);
$result = array_slice($cities, -2, 1);
echo "从倒数第 2 个开始，截取第 1 个元素产生的数组: \n";
print r($result) ;
$result = array slice($cities, 0, 3);
echo "从索引 0 开始，截取 3 个元素产生的数组: \n";
print r($result) ;
echo "从索引 2 开始，截取到倒数第 1 个产生的数组: \n";
print_r(array_slice($cities, 2, -1));
echo "从索引 2 开始，截取到倒数第 1 个产生的数组: \n";
print r(array slice($cities, 2, -1, true));
?>
```

执行结果如下所示：

```
从索引为 2 的元素开始截取产生的数组:
Array
(
    [0] => 郑州
    [1] => 南京
    [2] => 杭州
    [3] => 长沙
    [4] => 青岛
)
从倒数第 2 个开始，截取第 1 个元素产生的数组:
Array
(
    [0] => 长沙
)
从索引 0 开始，截取 3 个元素产生的数组:
Array
(
    [0] => 北京
    [1] => 上海
    [2] => 郑州
)
从索引 2 开始，截取到倒数第 1 个产生的数组:
Array
(
```

```
    [0] => 郑州
    [1] => 南京
    [2] => 杭州
    [3] => 长沙
)
从索引 2 开始，截取到倒数第 1 个产生的数组：
Array
(
    [2] => 郑州
    [3] => 南京
    [4] => 杭州
    [5] => 长沙
)
```

2. array_splice()函数

array_splice()函数的功能与 array_slice()函数相似，但它是将目标数组中的一部分元素截取出来，目标数组本身只留下未截取的元素，其语法形式如下：

```
array array splice ( array &$input , int $offset [, int $length [, array
$ replacement ]] )
```

它的$offset 与$length 参数与 array_slice()函数相同。该函数的可选参数$replacement 是一个数组，如果设置了此参数，则会使用此数组中的元素取代目标数组中被去掉的那部分。

例如，下面的代码演示了如何使用 array_splice()函数对$cities 数组进行截取操作。

```
<?php
$cities=array("北京","上海","郑州","南京","杭州","长沙","青岛");
$result = array_splice($cities, 5); /从索引为 5 的元素开始截取
echo "截取的元素有: \n";
print r($result);
echo "截取之后\$cities 数组的内容为: \n";
print_r($cities);                    //输出截取后$cities 中保留的元素

$newcities=array("深圳","广州");
$result = array_splice($cities, 2, 2, $newcities);
                                     //用$newcities 替换$cities 中的内容再截取
echo "替换后\$cities 数组的内容为: \n";
print r($cities);
echo "替换后截取的元素有: \n";
print r($result);
?>
```

执行结果如下所示：

```
截取的元素有:
```

```
Array
(
    [0] => 长沙
    [1] => 青岛
)
截取之后$cities 数组的内容为:
Array
(
    [0] => 北京
    [1] => 上海
    [2] => 郑州
    [3] => 南京
    [4] => 杭州
)
替换后$cities 数组的内容为:
Array
(
    [0] => 北京
    [1] => 上海
    [2] => 深圳
    [3] => 广州
    [4] => 杭州
)
替换后截取的元素有:
Array
(
    [0] => 郑州
    [1] => 南京
)
```

5.6.2 合并数组

数组的合并要比截取复杂，因为在合并时，多个数组中可能出现相同的“键-值”对，因此如何处理它们是合并的关键。PHP 提供了 array_combine()、array_merge()和 array_merge_recursive()函数来处理不同情况下的合并操作。

1. array_combine()函数

array_combine()函数可以将两个大小相等且不为空的数组组合成一个数组，其语法形式如下：

```
array array_combine(array $keys, array $values)
```

两个数组的组合方式为，将第一个数组的值作为新数组的键，而将第二个数组的值作为新数组的值。最后返回新数组，如果两个数组的单元数不同或者数组为空时返回 false。

【实践案例 5-15】

假设有两个保存颜色的数组，其中$colors_en 数组保存的是英文，$colors_cn 数组保存的是中文。现在要求，创建一个新数组并且使用英文作键、中文作为值。

下面是使用 array_combine()函数的实现代码：

```
<?php
$colors_en=array("red","yellow","blue","green","black");
                                                //此数组将作为新数组的键
$colors_cn=array("红色","黄色","蓝色","绿色","黑色");  //此数组将作为新数组的值
$result = array_combine($colors_en, $colors_cn);    //进行合并
print_r($result);
?>
```

执行结果如下所示：

```
Array
(
    [red] => 红色
    [yellow] => 黄色
    [blue] => 蓝色
    [green] => 绿色
    [black] => 黑色
)
```

2. array_merge()函数

array_merge()函数可以将多个数组的元素合并起来，其语法形式如下：

```
array array_merge(array $input_array1, array $input_array2 [, ..., array
$input_arrayN])
```

合并时 array_merge()函数会将后面所有数组中的元素都依次附加到第一个数组的末尾，最后返回新生成的数组。

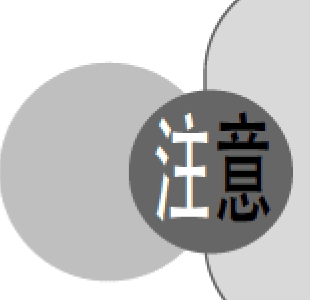
【实践案例 5-16】

假设，现在有 3 个保存成绩的数组。现在要求合并到一个数组并输出内容。使用 array_merge()函数实现代码如下：

```
<?php
$scores1 = array(100,'数学'=>98 , 'netmini' => 89);     //第一个数组
$scores2=array('itzcn'=>100,netmini => 98,'英语',71);   //第二个数组
$scores3=array(60,'语文'=>120,'itzcn' => 90);           //第三个数组
$result = array_merge($scores1, $scores2,$scores3);     //合并数组
print_r($result);                                       //输出数组的内容
?>
```

执行结果如下所示：

```
Array
(
    [0] => 100
    [数学] => 98
    [netmini] => 98
    [itzcn] => 90
    [1] => 英语
    [2] => 71
    [3] => 60
    [语文] => 120
)
```



该函数在附加时，如果数组元素使用的是数字键，则直接附加此元素，并使用默认值重新生成键；如果使用的是关联键（即非数字键），则会先在前面的元素中查找是否已存在此关联键，如果存在，则替换该键所对应的值，如果不存在，则附加此元素。

3. array_merge_recursive()函数

array_merge_recursive()函数的功能与 array_merge()函数相似，不同的是当存在相同的关联键时，array_merge()函数将使用后续附加的值替换原来的值；而 array_merge_recursive()函数则将该键作为一个数组名，将该关联键所对应的多个值作为此数组的元素。

array_merge_recursive()函数的语法形式如下：

```
array array_merge_recursive(array $input_array1, array $input_array2 [, ...,
array $input_arrayN])
```

例如，下面代码演示了使用 array_merge_recursive()函数合并数组$scores1 和数组$scores2 到新数组$result 中，然后输出新数组$result 的内容。

实现代码如下所示：

```
<?php
$scores1 = array('zhht' =>92,100,'英语', '数学' , 'somboy' => 89);
                                                //数组$scores1
$scores2=array(100,'zhht'=>98,somboy=> 58,71);      //数组$scores2
$result = array_merge_recursive($scores1, $scores2);    //合并数组
print_r($result);
?>
```

执行结果如下所示：

```
Array
(
    [zhht] => Array
        (
```

```
            [0] => 92
            [1] => 98
        )
    [0] => 100
    [1] => 英语
    [2] => 数学
    [somboy] => Array
        (
            [0] => 89
            [1] => 58
        )
    [3] => 100
    [4] => 71
)
```

5.7 数组排序

除了截取和合并数组之外，在计算机编程实践中数组的排序也是经常应用的操作。PHP支持的数据排序方式很多，像按键排序、按值排序、关联排序或者自定义排序等。本节将详细介绍这些排序的实现方式。

5.7.1 按值排序

很多排序算法都属于按值排序，例如冒泡排序、插入排序和选择排序等。即将数组中所有元素的值作为排序依据，然后按照升序或者降序进行排列。

PHP 提供了 4 个函数实现数组的按值排序：sort()、rsort()、natsort()和 natcasesort()，下面详细介绍它们。

1. sort()函数

sort()函数可以对目标数组进行排序，其排序方式是按元素值由低到高进行的，其语法形式如下：

```
bool sort ( array &$array [, int $sort_flags ] )
```

排序成功后返回 true，否则返回 false。$sort_flags 可选参数有如下值。

- **SORT_NUMBERIC　按数值排序，有利于对整数或浮点数排序。**
- **SORT_REGULAR　按自然顺序进行排序。**
- **SORT_STRING　按相应的 ASCII 值进行排序，适用于字符串元素。**

下面代码演示了如何对整型和字符串数组使用 sort()函数进行排序。

```
<?php
$scores = array(64,78,51,86,90);            //整型数组$scores
```

```
sort($scores);                                   //对$scores 进行排序
print_r($scores);                                //输出结果
$words = array('Apache', 'Web', 'Parent', 'Happy', 'Down');
                                                 //字符串数组$words
sort($words,SORT_STRING);                        //对$words 进行排序
print_r($words);                                 //输出结果
?>
```

执行结果如下所示：

```
Array
(
    [0] => 51
    [1] => 64
    [2] => 78
    [3] => 86
    [4] => 90
)
Array
(
    [0] => Apache
    [1] => Down
    [2] => Happy
    [3] => Parent
    [4] => Web
)
```



对含有混合类型值的数组使用 sort()排序可能会产生不可预知的结果。

2. rsort()函数

rsort()函数的功能与 sort()函数相类似，只不过 rsort()函数以相反的顺序对数组元素进行排序，其语法形式如下：

```
bool rsort ( array &$array [, int $sort_flags ] )
```

各个参数的含义均与 sort()函数相同，这里就不再介绍。

例如，对 sort()函数中的实例使用 rsort()函数进行降序排列，将看到如下的执行结果：

```
Array
(
    [0] => 90
    [1] => 86
    [2] => 78
    [3] => 64
```

```
    [4] => 51
)
Array
(
    [0] => Web
    [1] => Parent
    [2] => Happy
    [3] => Down
    [4] => Apache
)
```

3. natsort()函数

在学习 natsort()函数之前，首先来看一个数组排序代码，如下所示：

```
$pics=array("picture24","picture1","picture12","picture2");
sort($pics,SORT_STRING);          //对$pics 按字符串进行排序
print_r($pics);                   //输出排序后数组
```

最后的排序结果如下：

```
Array
(
    [0] => picture1
    [1] => picture12
    [2] => picture2
    [3] => picture24
)
```

也就是说，字符串中的数字并没有按我们预想的排序。这时，可以使用 natsort()函数来解决这样的问题，natsort()函数的语法形式如下：

```
void natsort(array $target_array)
```

现在将示例中的“sort($pics,SORT_STRING);”替换为“natsort($pics)”重新排序，执行后的结果如下所示：

```
Array
(
    [1] => picture1
    [3] => picture2
    [2] => picture12
    [0] => picture24
)
```

4. natcasesort()函数

natcasesort()函数在功能上与 natsort()函数差不多，只是 natcasesort()函数在排序时不区

分大小写，其语法形式如下：

```
bool natcasesort ( array &$array )
```

例如，下面的代码说明了使用 natcasesort()函数和 natsort()函数排序结果的差异：

```
<?php
$files=array("file2.txt","File18.jpg","FILE5.gif","file14.bmp","FILE4.p
ng");
natsort($files);                        //使用 natsort()函数排序
echo "使用 natsort()函数排序后的结果: \n";
print r($files);
echo "natcasesort()函数排序后的结果: \n";
natcasesort($files);                    //使用 natcasesort()函数排序
print r($files);
?>
```

执行结果如下所示：

```
使用 natsort()函数排序后的结果:
Array
(
    [4] => FILE4.png
    [2] => FILE5.gif
    [1] => File18.jpg
    [0] => file2.txt
    [3] => file14.bmp
)
natcasesort()函数排序后的结果:
Array
(
    [0] => file2.txt
    [4] => FILE4.png
    [2] => FILE5.gif
    [3] => file14.bmp
    [1] => File18.jpg
)
```

5.7.2　按键排序

数组的按键排序要比按值排序简单，主要通过 ksort()函数和 krsort()函数实现。

1. ksort()函数

ksort()函数可以按键对数组进行排序，其语法形式如下：

```
bool ksort ( array &$array [, int $sort_flags ] )
```

$array 表示要排序的数组，可选参数$sort_flags 与 sort()函数中的作用相同，这里就不再介绍。

假设，在会员数组中使用编号作为键，会员名称作为值。现在要按会员编号进行排序，使用 ksort()函数的实现代码如下：

```
<?php
$users=array(30=>"mary",27=>"xiaoqiang",39=>"laoshu",21=>"java163",25=>
"murphy");
ksort($users);                    //对$users 数组按键排序
print_r($users);                  //输出排序后的内容
?>
```

排序结果如下所示：

```
Array
(
    [21] => java163
    [25] => murphy
    [27] => xiaoqiang
    [30] => mary
    [39] => laoshu
)
```

2. krsort()函数

krsort()函数在功能上与 ksort()函数相同，只不过 krsort()函数以相反的顺序对数组进行排序，其语法形式如下：

```
bool krsort ( array &$array [, int $sort_flags ] )
```

例如，同样是前面创建的会员数组$users，现在使用 ksort()函数对它按键进行降序排列，将看到如下排序结果：

```
Array
(
    [39] => laoshu
    [30] => mary
    [27] => xiaoqiang
    [25] => murphy
    [21] => java163
)
```

5.7.3　关联排序

前面介绍的 sort()函数可以按值对数组排序，但是排序后元素的键被重新生成，有时候这种行为会影响应用程序。不过，PHP 中也提供了排序保持“键-值”对的函数。

1. asort()函数

使用 sort()函数排序数组时，数组中对应各值的键会重新生成。例如，下面是用 sort()函数排序的示例代码：

```
<?php
//创建带键的数组
$scores = array('murphy' => 92,'lelei' => 90, 'chengj' => 85,"zhht"=> 100,
'forever' => 89);
sort($scores);                //使用 sort()函数排序
print_r($scores);             //输出排序结果
?>
```

其执行结果如下所示：

```
Array
(
    [0] => 85
    [1] => 89
    [2] => 90
    [3] => 92
    [4] => 100
)
```

对比结果中的“键-值”对与代码中$scores 数组的“键-值”对，会发现“键-值”对没有被作为一个整体进行排序，而是另外生成数值类型的默认键。而有时候，我们希望在值被排序的同时，其键也跟随其一起移动。

使用 asort()函数即可实现这一需求，asort()函数的语法形式如下：

```
bool asort ( array &$array [, int $sort_flags ] )
```

函数中的参数与 sort()函数相同，这里就不再介绍。下面使用 asort()函数对前面的$scores 数组进行排序，排序的结果如下所示：

```
Array
(
    [chengj] => 85
    [forever] => 89
    [lelei] => 90
    [murphy] => 92
    [zhht] => 100
)
```

从结果中可以看到，asort()函数在实现数组升序排列的同时保留了值的键。

2. arsort()函数

arsort()函数在功能上与 asort()函数相同，只不过 arsort()函数以相反的顺序对数组进行

排序，其语法形式如下：

```
bool arsort ( array &$array [, int $sort_flags ] )
```

函数中的参数与 sort()函数相同，这里就不再介绍。

例如，下面使用 arsort()函数对前面的$scores 数组进行降序排列，代码如下所示：

```
<?php
$scores = array('murphy' => 92,'lelei' => 90, 'chengj' => 85,"zhht"=> 100,
'forever' => 89);
arsort($scores);                        //使用 arsort()函数排序
print r($scores);
?>
```

执行结果如下所示：

```
Array
(
    [zhht] => 100
    [murphy] => 92
    [lelei] => 90
    [forever] => 89
    [chengj] => 85
)
```

从上述的结果中可以看到，与使用 asort()函数的结果刚好相反，它在实现数组降序排列的同时保留了值的键。

5.7.4 级联排序

级联排序是指以一个主数组为依据同时对多个数组进行排序。PHP 使用 array_multisort()函数实现级联排序，语法形式如下：

```
bool array_multisort(array $array [, mixed $org1 [, mixed $org2]] [, array
$arg ...]])
```

在该函数中除第一个参数必须是一个数组外，其他参数可以是数组或者是下面两种排序标志。

排序顺序标志：

- ❑ **SORT_ASC　按照上升顺序排序。**
- ❑ **SORT_DESC　按照下降顺序排序。**

排序类型标志：

- ❑ **SORT_REGULAR　将元素按照通常方法比较。**
- ❑ **SORT_NUMERIC　将元素按照数值比较。**
- ❑ **SORT_STRING　将元素按照字符串比较。**

array_multisort()函数主要用于多个数组之间的级联排序，其中第一个数组为主要数组，后面的数组中的元素与第一个数组中的元素一一对应。这样当第一个数组中的元素位置发生变化后，后面的数组中的元素也跟着变换位置。

例如，下面的代码创建了 3 个数组$letters、$words 和$cities。然后，使用 array_multisort()函数以$letters 数组中的元素为依据作排列，并同时关联到其他 2 个数组。

```
<?php
$letters=array("A","C","D","B");            //创建一个数组，以它作为排序依据
$words=array("apache","car","days","book");//创建一个关联数组
$cities=array("北京","上海","广州","深圳");   //创建一个关联数组
array multisort($letters, SORT DESC,SORT STRING, $words, $cities);
                                            //按$letters 数组进行排列
echo "\$letters 排序后的结果是: \n";
print_r($letters);                          //输出排序后$letters 数组的内容
echo "\$words 排序后的结果是: \n";
print_r($words);                            //输出排序后$words 数组的内容
echo "\$cities 排序后的结果是: \n";
print r($cities);                           //输出排序后$cities 数组的内容
?>
```

如上述示例，第一个数组$letters 中有 4 个字符串，后面 2 个数组中也都有 4 个元素，这样就与$letters 数组中的元素一一对应。接下来使用 array_multisort()函数对第一个数组进行降序排到，并将排序关联$words 和$cities 数组中。

最终执行结果如下所示：

```
$letters 排序后的结果是:
Array
(
    [0] => D
    [1] => C
    [2] => B
    [3] => A
)
$words 排序后的结果是:
Array
(
    [0] => days
    [1] => car
    [2] => book
    [3] => apache
)
$cities 排序后的结果是:
Array
(
    [0] => 广州
```

```
        [1] => 上海
        [2] => 深圳
        [3] => 北京
    )
```



如果使用 array_multisort()函数对多个数组进行级联排序，则必须保证多个数组中的元素个数一样，否则该函数不会执行任何操作。

5.7.5 自定义排序

虽然 PHP 提供了众多的排序方式和排序函数，但是在实际应用时仍然可能无法满足排序要求。例如，希望对数组按字符串长度排序，按日期格式排序，或者按数字的奇偶性排序等。

PHP 提供了一个名为 usrot()的函数，允许用户编写自定义比较算法对数组进行排序。该函数语法形式如下：

```
bool usort ( array &$array , callback $cmp_function )
```

其中，$array 参数是要排序的数据，$cmp_function 参数是自定义排序函数的名称。$cmp_function 函数必须在第一个参数被认为小于、等于或大于第二个参数时分别返回一个小于、等于或大于零的整数。

【实践案例 5-17】

假设一个数组里面保存的全部是整数。现在要求对整数进行排序并输出，排序规则是先奇数后偶数，然后按数字从大到小降序排列。

由于 PHP 没有提供类似的排序函数。要实现这个效果就需要自定义一个函数，再通过 usort()函数进行调用。实现代码如下所示：

```
<?php
function Compare ($str1, $str2){                //自定义排序函数
    if(($str1 %2==0)&&($str2%2==0))             //第 1 个数和第 2 个数都是偶数
    {
        if($str1>$str2) return -1;
        else return 1;
    }
    if($str1%2==0) return 1;                    //判断第 1 个数是否偶数
    if($str2%2==0) return  -1;                  //判断第 2 个数是否偶数
    return ($str2 > $str1) ? 1:-1 ;             //判断两个数的大小关系
}
$scores = array(64,78,51,83,97,60,87,91,100,17,62);     //创建数组
usort($scores, 'Compare');                      //调用自定义排序函数
echo "排序后的数组为: \n";
foreach ($scores as $score) echo "$score ";//输出排序后数组的内容
```

```
?>
```

如上述代码，在文件中自定义一个 Compare()函数，在该函数中对两个参数的奇偶性和大小进行判断，并返回 1 或者-1。然后在 usort()函数中调用该函数对$scores 数组进行排序，排序后的结果如下所示：

```
排序后的数组为:
97 91 87 83 51 17 100 78 64 62 60
```

从输出结果中可以看到，Compare()函数对$scores 数组中的元素实现了预期的排序。

5.8 项目案例：制作查看教程页面

通过本章前面的学习，相信读者一定掌握了 PHP 对数组的各种操作。那么本次案例将综合这些知识制作一个教程网站的查看页面。由于还没有学习如何在网页中保存数据，以及从数据库中读取数据的内容。因此在这里使用本章介绍的数组作为数据源保存页面所需各种数据。然后，利用本章介绍的数组函数对它们进行排序和遍历，并输出到页面上。

【实例分析】

为了实现网站页面的动态效果，这里将所有数据均保存到数组中，主要包括如下内容。

- ❑ **网站导航菜单数组（编号、菜单名称、链接地址）。**
- ❑ **网站底部菜单数组（菜单名称、链接地址）。**
- ❑ **当前位置数组。**
- ❑ **新闻数组（新闻编号、标题、来源、浏览次数、发布时间、内容）。**
- ❑ **配置数组（网站名称、备案号、网站地址、总访问量）。**

（1）打开设计好的教程网站页面，将它另存为 index.php。

（2）在页面的顶部使用 PHP 创建第一个数组，该数组保存的是网站的导航菜单，具体代码如下所示：

```
<?php
$menus=array(
        5=>array("name"=>"教育学院","url"=>"school.html"),
        2=>array("name"=>"学术论坛","url"=>"bbs.html"),
        3=>array("name"=>"技术文档","url"=>"blog.html"),
        6=>array("name"=>"娱乐空间","url"=>"space.html"),
        1=>array("name"=>"窗内首页","url"=>"index.html")
);
?>
```

可以看到$menus 实际上是一个二维数组，这里定义了 5 个元素，每个元素都带有一个整数键表示菜单编号。元素中的 name 表示菜单名称、url 表示菜单的链接地址。

（3）在页面显示导航菜单位置读取$menus 中的数据并显示，但在这之前需要先对它进行排序。排序代码如下所示：

```
<ul>
  <?php
  ksort($menus);            //调用 ksort()函数对$menus 执行按键值排序
  foreach ($menus as $menu)
  {
    echo "<li><a href='".$menu['url']."'>".$menu['name']."</a></li>";
  }
  ?>
</ul>
```

（4）在上述代码中，使用了 PHP 混合 HTML 的嵌入方式。执行后将会在页面生成如下的 HTML 代码：

```
<ul>
   <li><a href='index.html'>窗内首页</a></li>
   <li><a href='bbs.html'>学术论坛</a></li>
   <li><a href='blog.html'>技术文档</a></li>
   <li><a href='school.html'>教育学院</a></li>
   <li><a href='space.html'>娱乐空间</a></li>
 </ul>
```

（5）创建一个普通数组保存当前页面在网站中的位置，其中每个元素表示一个层次。代码如下所示：

```
<?php
$position=array("首页","新手上路","什么是窗内网");
?>
```

（6）如上述代码所示，$position 数组非常简单，不带键也没有嵌套。所以在这里使用 for 语句来进行遍历来输出，代码如下所示：

```
  <div id="sHeader">
  <?php
  $min = key($position);                        //获取第一个键
  $length = count($position);                   //获取键的数量
  $max = $min + $length - 1;                    //计算键的最大值
  for($i = $min; $i <= $max; $i ++){
    echo "$position[$i]  &gt;";                 //循环输出
  }
  ?>
  </div>
```

（7）接下来创建一个用于保存教程信息的数组$news，包括教程编号、标题、来源、浏览次数、发表日期和详细内容。本示例中使用的定义代码如下所示：

```
<?php
$news=array(
```

```
            'id'=>3,
            'title'=>"窗内网",
            'source'=>"教程网",
            'views'=>349493,
            'pubdate'=>"2012-8-4",
            'text'=>"窗内网 (www.itzcn.com) 是一个基于 Web 2.0 的真实社区...."
    );
    ?>
```

如上述代码所示，在$news 数组中每个元素都指定了键。这样使键和值具有关联性，键相当于数据库中表的列名，值就是列的数据。

（8）编写代码在页面中显示$news 中的教程信息，在这里要注意使用键的方式获取数组内的数据，实现代码如下所示：

```
<div class="content">
  <h1><?php echo $news['title']?></h1>
  <h2>来源: <?php echo $news['source']?> | 浏览次数: <?php echo $new'views']?>
  次 | 发布时间: <?php echo $news['pubdate']?></h2>
  <p> <?php echo $news['text']?><br />
  </p>
</div>
```

（9）这一步来实现教程页面的底部导航和网站信息的信息。这需要使用两个数组，它们的定义如下所示：

```
<?php
$foots=array(
        array("关于我们","about.html"),array("免责声明","licence.html"),
        array("广告合作","ad.html"),array("在线帮助","help.html"),
        array("支付方式","pay.html"),array("加入我们","join.html"),
);
$config=array('name'=>"窗内网教程频道",'url'=>"www.itzcn.com",'bean'=>"豫
ICP 备 000000 号",'visitors'=>494834);
?>
```

（10）可以看到$foots 也是一个二维数组，但里面的子数组没有带键，所以这里使用 foreach 进行遍历输出菜单名称和链接地址。$config 数组则是使用键来指定配置名称，使用值指定配置的结果。如下所示是这两个数组的遍历代码：

```
<P class="foot">
<?php
foreach ($foots as $menu)
{
  echo "<a href='".$menu[1]."'>".$menu[0]."</a> | ";
}
?>
```

```
  </P>
<P class="foot"><EM>Copyrights Reserved 2012</EM> <?php echo
$config['name']?>(<EM><?php echo $config['url']?></EM>) <br> <EM><?php echo
$config['bean']?> 流量: <?php echo $config['visitors']?></EM>
</P>
```


（11）至此，在教程页面中的动态数据显示就已经完成了。运行 index.php 将看到如图 5-4 所示的运行效果。



图 5-4　查看教程页面运行效果

在实例中共使用了 5 个数组。它们的共同实现思路是，先使用数组来定义所需的数据，然后对数组进行操作（像排序、统计个数、汇总），再遍历数组输出到页面。

5.9　习题

一、填空题

（1）下列代码执行后$sites 数组中有__________个元素。

```
<?php
$sites[10]="窗内网";
$sites["itzcn"]="计算机教程网";
$sites[]="汇智科技";
$sites[58]=2012;
?>
```

（2）下列代码执行后的输出是__________。

```
<?php
$ary[]=null;
if(is array($ary))
    echo "true";
else
    echo "false";
?>
```

（3）下列代码执行后的输出是__________。

```
<?php
$colors=array("RED","blue","green","black","yellow","white");
if (in array("red", $colors))
   echo " true ";
 else
    echo "false";
?>
```

（4）在下面代码空白处填写合适的代码，使程序可以向$students 数组的末尾添加 2 个元素。

```
<?php
$students=array("祝红涛","侯艳书");
            ($students,"刘强","张华");
?>
```

（5）假设要合并两个数组，要求将第一个数组的值作为新数组的键，而将第二个数组的值作为新数组的值。实现时使用__________函数最合适。

二、选择题

（1）下列关于 PHP 中数组的描述，正确的是。__________

A．数组对应的是 Array 类，使用 new 进行创建

B．数组的键在创建时可以不指定，默认从 1 开始

C．数组的键必须是数字类型

D．数组可以直接赋值，或者用 array()函数，键也可以是字符串类型

（2）下列对$ary 数组的操作不正确的是。__________

A．$ary[]=null;

B．$ary['58m']="58.com";

C．$ary=array();

D．$ary[]=new array();

（3）要输出数组的内容，应该使用下列哪个函数。__________

A．print_r()

B．echo()

C．printf()

D．print_array()

（4）下列代码执行后$result 的结果是__________。

```
<?php
$cities=array('北京','广州','上海',
            'henan'=>array('郑州','安阳'),
            'zhejiang'=>array('温州','杭州')
            );
$result=count($cities,1);
?>
```

A．6

B．7

C．8

D．9

（5）假设要统计$ary 数组中各元素出现的频率，下列代码正确的是。__________

A．array_unique($ary)

B．array_count_values($ary)

C．array_push($ary)

D．array_keys($ary)

（6）下列函数不能实现从数组中提取元素功能的是。__________

A．array_diff()

B．array_diff_assoc()

C．array_intersect()

D．array_search()

（7）下列关于数组排序的描述，不正确的是__________。

A．可以按数组的值进行升序或者降序排列

B．可以按数组的键进行升序或者降序排列

C．可以实现多个数组同时排序

D．可以实现多维数组的排序

三、上机练习

1．编写冒泡排序

本次上机要求读者不借用任何函数，实现一个整型数组的冒泡排序。冒泡排序算法的规则是：搜索整个数组，比较相邻元素，如果两者的相对大小次序不对，则交换它们，其结果是最大值“像水泡一样”移动到数组的最后一个位置上，这也是它在最终完成排序的数组中合适的位置。然后再次搜索数组，将第二大的值移动至倒数第二个位置上，重复该过程，直至将所有元素移动到正确的位置上。

5.10 实践疑难解答

5.10.1 如何返回数组中相同键值的键名



如何返回数组中相同键值的键名
网络课堂：http://bbs.itzcn.com/thread-19300-1-1.html

【问题描述】：请问在 PHP 的数组中如何能返回相同键值的键名啊？

假设我的数组内容如下：

```
$array=array("水费"=>"每月","宽带费"=>"每季度","电费"=>"每月","有线费"=>"每年
","物业费"=>"每年","话费"=>"每月","停车费"=>"每季度");
```

最终想得到类似下面的结果：

```
每月：水费 电费 话费
每季度：宽带费 停车费
每年：有线费 物业费
```

我应该怎么做啊？

【解决办法】：这个题目不难，关键看你对数组的基础掌握怎么样。其实只要使用两个遍历即可实现，代码如下：

```
$array=array("水费"=>"每月","宽带费"=>"每季度","电费"=>"每月","有线费"=>"每年"
业费"=>"每年","话费"=>"每月","停车费"=>"每季度");
$new array = array();
foreach($array as $key =>$val){
    $new_array[$val][] = $key;
}
foreach($new array as $key=>$val){
    echo $key,': ',implode(' ',$val),'<br />';
}
```

另外，这个其实就是查找相同的键，也可以使用 PHP 的数组函数来实现。这种方式的代码如下：

```
$key=array_keys($array,'每月');
$key1=array_keys($array,'每季度');
$key2=array keys($array,'每年');
print_r($key);
print r($key1);
print_r($key2);
```

5.10.2 怎样把同一数组中相同的键值合并为一个



怎样把同一数组中相同的键值合并为一个
网络课堂：http://bbs.itzcn.com/thread-19301-1-1.html

【问题描述】：定义了一个数组，包含的内容如下：

```
$ary=array("移动","联通","电信","3G","电信");
```

现在要把重复出现的“电信”合并为一个，也就是相同的值都合并。不知道应该怎么解决，有没有类似的函数呢？

【解决办法】：PHP 开发人员早就考虑到这种情况，并提供了一个 array_unique()函数。看看下面的示例：

```
<?php
$a=array("a"=>"Cat","b"=>"Dog","c"=>"Cat");
print r(array unique($a));
?>
```

输出是：Array ([a] => Cat [b] => Dog)。很简单，知道该怎么做了吧。

第6章 字符串处理

文字是记录和传播信息的重要载体，因此经常需要在程序中处理大量文字。在 PHP 中文字信息被保存为字符串来处理，同时字符串又称字符数组，是 PHP 中非常重要的数据类型之一。

本章将详细介绍 PHP 支持的字符串操作，像创建多行字符串、获取字符串长度和单词数量、大小写替换、去除多余字符、分隔字符串，以及字符串的填充和替换等。

本章学习要点：

- 理解 PHP 字符串与数组的对应关系
- 熟悉创建字符串的 3 种方法
- 掌握统计字符串长度、出现次数和单词数量的方法
- 掌握字符串的大小写替换
- 掌握去除字符串多余字符的操作
- 熟悉字符串查找和比较操作
- 掌握对子字符串的各种处理

6.1 创建字符串

字符串是 PHP 标量数据类型之一，第 2 章简单介绍了字符串的创建方式，本节将详细介绍创建字符串的具体内容。

6.1.1 字符串与数组的转换

PHP 中的字符串其实是由若干字符组成的集合，包含的字符可以是大写字母、小写字母、汉字、数字以及*、#等特殊符号。而且 PHP 对于字符串的长度没有限制，因为字符串长度的限制只与运行 PHP 程序的计算机内存大小有关，与 PHP 本身无关。

例如，下面的代码创建了一个包含“hello php”字符串的变量$str。

```
$str="hello php";
```

由于字符串的每个字符都是连续的，所以也可以把字符串作为数组来处理。图 6-1 所示为两者之间的转换示意图。

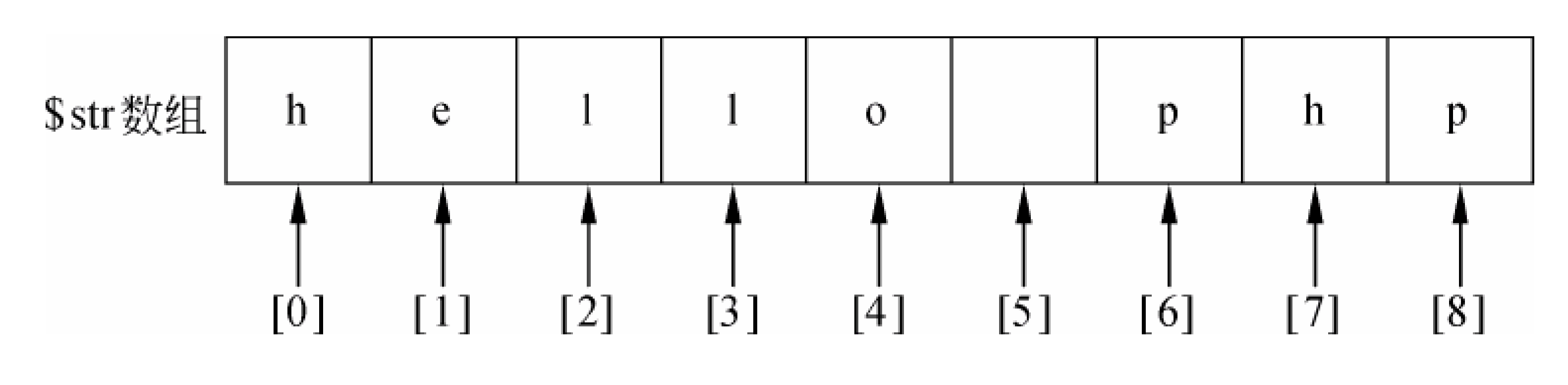


图 6-1 字符串与数组的转换示意图

【实践案例 6-1】

创建一个案例演示字符串的标准处理方式和数组处理方式，示例代码如下所示：

```
<?php
$str="PHP is very simple";  //创建一个字符串变量$str，赋值为 PHP is very simple
echo $str;                  //输出：PHP is very simple
echo $str[0];               //输出$str 的第 1 个字符，结果为：P
echo $str[4];               //输出$str 的第 5 个字符，结果为：i
echo $str[7];               //输出$str 的第 8 个字符，结果为：v
?>
```

可以看到，在上述代码中使用数组的输出结果和使用字符串的输出结果是一样的。在这里使用中括号来表示字符串的数组形式输出，可能会和数组产生二义性。因为数组也使用中括号。因此，在 PHP 中建议使用大括号{}代替在这里的中括号[]，以消除语法上的二义性，即使用$str{0}替换$str[0]。

6.1.2 双引号创建

定义一个字符串最简单的方法是用双引号（"）把它括起来。这种方式定义的字符串，PHP 会对其中的特殊字符（像双引号、反斜线、美元符号）进行解析，另外，如果有变量也会先执行，再使用值输出。

【实践案例 6-2】

创建一个案例演示如何使用双引号定义单行字符串、多行字符串、包含特殊字符和变量的字符串，示例代码如下：

```
<?php
echo "我的未来不是梦，我的心跟着希望在动";          //定义一个普通单行字符串
echo "恭喜你获得游戏的二等奖。请登录下面的网址领取：
        http://www.itzcn.com/game/get?id=3&k=djksdo93c8033
   兑奖截止日期：2012 年 12 月 31 日
   最终解释权归无聊所有
     ";                                            //使用双引号定义多行字符串
echo "/* this function is outputs or returns a parsable string representation
of a variable */";
echo "\I don't want the dead-end job";             //包含单引号的字符串
echo "\"She really takes a shine to you\",she said.";
                                                   //包含双引号的字符串
```

```
$city="郑州";
echo "$city 是一个古老的城市";                    //带变量的字符串，变量会解析值
?>
```

上述代码执行后，将看到如下的字符串输出结果：

```
我的未来不是梦，我的心跟着希望在动
恭喜你获得游戏的二等奖。请登录下面的网址领取：
    http://www.itzcn.com/game/get?id=3&k=djksdo93c8033
    兑奖截止日期：2012 年 12 月 31 日
    最终解释权归无聊所有
/* this function is outputs or returns a parsable string representation of
a variable */
\I don't want the dead-end job
"She really takes a shine to you",she said.
郑州 是一个古老的城市
```

6.1.3 单引号创建

除了双引号之外，还可以使用单引号创建一个字符串，即使用单引号（'）把字符串括起来，这种方式在输出一个单引号时需在它的前面加个反斜线（\）。如果在单引号前或在字符串的结尾处想要输出反斜线，输入两个反斜线（\）。而在其他任何字符前加反斜线，反斜线将会被直接输出。

【实践案例 6-3】

创建一个案例演示如何使用单引号定义单行字符串。例如，下面的这些字符串定义方式都是正确的：

```
<?php
echo '请仔细阅读说明书，并按说明书使用或者在药师指导下购买和使用\n';
                                             //标准的字符串形式
echo '产品名称：润洁
       专业名称：复方硫酸软骨素滴眼液
     适应症：眼疲劳、眼干燥症
';                                           //使用单引号定义的多行字符串
echo 'Let\'s say that it\'s true.';          //带单引号的字符串.
echo '"Yes," said Ford.';                    //带双引号的字符串
echo '床/沙发/衣柜/宜家/办公家具';           //带斜线的字符串
echo '早教\玩具\戏水';                       //带反斜线的字符串
$age=10;
echo '我今天 $age 岁了';                     //带变量的字符串，变量不会解析
?>
```

如上述代码所示，单引号与双引号定义字符串的方式相同，唯一不同的是单引号中的变量不会解析和运行。上述代码执行后，将看到如下的字符串输出结果：

```
请仔细阅读说明书，并按说明书使用或者在药师指导下购买和使用\n
产品名称：润洁
        专业名称：复方硫酸软骨素滴眼液
        适应症：眼疲劳、眼干燥症
Let's say that it's true.
"Yes," said Ford.
床/沙发/衣柜/宜家/办公家具
早教\玩具\戏水
我今天 $age 岁了
```

6.1.4 定界符创建

定界符适用于创建多行字符串，它需要在字符串前添加定界符前缀“<<<”，后跟一个定界标识符；接下来在新行中定义字符串的内容，可以使用双引号、单引号、特殊字符和变量；最后使用定界标识符作为字符串的结束标志。

例如，下面使用定界符方式创建了一个字符串：

```
<?php
echo <<<WEL
欢迎光临欣悦快餐店
今日优惠菜单:
麻婆豆腐、剁椒鱼头、香米海瓜、酥皮酱鸭
WEL                         //钓竿字符串，定界标识符为 WEL
?>
```

上述代码使用 WEL 作为字符串的定界标识符。使用时要注意定界标识符在结束时必须在一行的开始位置，而且命名也要遵守一定的规则：只能包含字母、数字和下划线，并且不能用数字和下划线作为开头。

下面的示例代码演示了如何在定界符字符串中使用双引号和变量：

```
<?php
echo <<<my
"i am a Chinese" she said.
and what's your name?
my;                         //双引号和单引号都不需要转义
$price=39.0;
echo <<<Product
您购买商品的售价为: $price
使用的是变量为\$price
Product;                    //变量前需要加斜线\，否则会被执行
?>
```

6.2 统计字符串

创建字符串之后便可以对它进行操作。在众多的操作之中对字符串进行统计操作是最

简单，也是最常用的。PHP 为此提供了 3 个函数：strlen()、count_chars()和 str_word_count()。

6.2.1　统计字符串长度

PHP 的 strlen()函数用于统计字符串长度，其语法格式为：

```
int strlen ( string $string )
```

该函数返回值为整型，表示字符串$string 的长度。如果返回值为 0 表示该字符串为空。

【实践案例 6-4】

在会员系统中对会员的密码有这样的规则：密码长度必须大于 6 位且小于 12 位。因为密码太短容易被破解，太长的话不容易记住。

这就需要根据给出的密码判断长度是否在范围之内，并给出相应的提示。使用 strlen()函数的实现代码如下所示：

```
<?php
function CheckPasswordLenght($pass){
    if(strlen($pass)>=6 && strlen($pass)<=12)
        echo "密码符合策略。";
    elseif(strlen($pass)>12 )
        echo "不符合密码策略。密码超过最大长度 12 位。";
    else
        echo "不符合密码策略。过于简单的密码不安全。";
}
CheckPasswordLenght(" 1 2 ");         //不符合密码策略。过于简单的密码不安全。
CheckPasswordLenght("1 3 5 7");       //密码符合策略。
CheckPasswordLenght("mynameiszhuhongtao");
                                      //不符合密码策略。密码超过最大长度 12 位。
?>
```

上述代码创建了一个 CheckPasswordLenght()函数用于判断指定的$pass 是否符合规则，并输出提示信息。在函数中使用 strlen()函数获取字符串$pass 的长度并进行比较，然后给出结果。在这里要注意，一个字符串中的空格同样也被计算在内。

6.2.2　统计字符出现频率

使用 count_chars()函数可以统计一个字符串中的字符出现频率，它的语法格式如下：

```
mixed count_chars ( string $string [, int $mode ] )
```

其中，$mode 是可选参数用于指定不同的计数模式，其默认值为 0，可选值如下。

- **0　返回一个关联数组，由所有字符作为键，该字符在原字符串中出现的次数作为值。**
- **1　与 0 相同，但只返回出现次数大于零的字符。**

- 2　与 0 相同，但只返回出现次数等于零的字符。
- 3　返回一个字符串，其中包含原字符串中能找到的所有字符，每个字符只显示一次。
- 4　返回一个字符串，其中包含原字符串中未使用的所有字符。

例如，下面的代码演示如何使用 count_chars()获取一个字符串中的字符总数：


```
<?php
$words="woo haa yeah";
foreach (count chars($words, 1) as $key=>$value) {
    echo "字符\"".chr($key)."\"共出现了".$value."次\n";
}
?>
```

执行结果如下所示：

```
字符" "共出现了 2 次
字符"a"共出现了 3 次
字符"e"共出现了 1 次
字符"h"共出现了 2 次
字符"o"共出现了 2 次
字符"w"共出现了 1 次
字符"y"共出现了 1 次
```

6.2.3　统计单词数量

str_word_count()函数用于统计一个字符串的单词数，它的语法格式如下：

```
mixed str_word_count ( string $string [, int $format = 0 [, string $charlist ]] )
```

该函数的$format 可选参数用于指定计数形式，其默认值为 0，可选值如下。

- **0　返回字符串中单词的总数。**
- **1　返回由字符串中所有单词组成的数组。**
- **2　返回一个关联数组，其中单词的所在位置为键，单词本身为值。**

$charlist 参数可以指定附加的字符串列表，其中的字符将被视为单词的一部分。例如，下面的代码演示如何使用 str_word_count()获取一个字符串中的字符总数：

```
<?php
$words="He was 25 years old";
$result=str word count($words);
echo "\$words 中共有单词： $result 个";
print r(str word count($words, 1));
print r(str word count($words, 2));
print r(str word count($words, 1, '25'));   //将 25 也作为一个单词
?>
```

执行结果如下所示：

```
$words 中共有单词: 4 个
Array
(
    [0] => He
    [1] => was
    [2] => years
    [3] => old
)
Array
(
    [0] => He
    [3] => was
    [10] => years
    [16] => old
)
Array
(
    [0] => He
    [1] => was
    [2] => 25
    [3] => years
    [4] => old
)
```

6.3 操作字符串内容

针对字符串的统计操作不会对字符串的原始内容进行任何处理。本节将介绍一些操作字符串内容的函数，像将所有小写替换为大写、去除首尾的特殊字符以及比较两个字符串的大小等。

6.3.1 大小写替换

字符串的大小写替换主要可分为如下几种情况，PHP 为每种情况都提供了处理函数。

- **将字符串中的所有字符都转换成大写。**
- **将字符串中的所有字符都转换成小写。**
- **将字符串的第一个字符转换成大写。**
- **将字符串中的所有单词的第一个字符都转换成大写。**

1. strtolower()函数

strtolower()函数可以将字符串$str 中的所有字符全部转换成小写，对于非字母的字符不受影响。语法格式如下所示：

```
string strtolower ( string $str )
```

例如，下面的代码演示了如何使用 strtolower()函数：

```
<?php
$str1="Make Money From Your Site";                  //定义原始字符串
echo "$str1 的小写结果: ".strtolower($str1);           //转换并输出
$str2="SELECT * FROM TABLE1 WHERE AGE=22 AND UID>18";
echo "\n$str2 的小写结果: ".strtolower($str2);
$str3="告诉我这到底是为什么? Tell Me Why?";
echo "\n$str3 的小写结果: ".strtolower($str3);
?>
```

执行结果如下所示：

```
make money from your site
select * from table1 where age=22 and uid>18
告诉我这到底是为什么? tell me why?
```

2. strtoupper()函数

strtoupper()函数的功能与 strtolower()函数相反，它可以将字符串$string 中的所有字符全部转换成大写，而非字母的字符不受影响。语法格式如下所示：

```
string strtoupper ( string $string )
```

例如，下面的代码演示了如何使用 strtoupper()函数：

```
<?php
$str1="show ads that relate to the content on your website";
                                                    //定义原始字符串
echo "$str1 的大写结果: ".strtoupper($str1);           //转换并输出
$str2="Business Solutions";
echo "\n$str2 的大写结果: ".strtoupper($str2);
$str3="歌曲名称: the day you went away";
echo "\n$str3 的大写结果: ".strtoupper($str3);
?>
```

上述代码都很简单，这里就不再解释，运行后的输出结果如下所示：

```
SHOW ADS THAT RELATE TO THE CONTENT ON YOUR WEBSITE
BUSINESS SOLUTIONS
歌曲名称: THE DAY YOU WENT AWAY
```

3. ucfirst()函数

ucfirst()函数可以将字符串$str 中的第一个字符（如果是字母）转换成大写，而非字母的字符不受影响。语法格式如下：

```
string ucfirst ( string $str )
```

【实践案例 6-5】

在书写英文时每个段落开始的首字母必须大写。下面使用 ucfirst()函数实现将字符串中的首字符转换为大写：

```
<?php
$str1="those customers you are looking for, are looking for you - on Google";
                                                        //定义原始字符串
echo "使用 ucfirst()函数之后为: ".ucfirst($str1);          //转换并输出
$str2="歌曲名称:  the day you went away";
echo "\n 使用 ucfirst()函数之后为: ".ucfirst($str2);
?>
```

执行后的结果如下所示：

```
使用 ucfirst()函数之后为: Those customers you are looking for, are looking for
you - on Google
使用 ucfirst()函数之后为: 歌曲名称:  the day you went away
```

4．ucwords()函数

ucwords()函数可以将字符串$str 中每个单词的第一个字符转换成大写，非字母字符不受影响。语法格式如下：

```
string ucwords ( string $str )
```

【实践案例 6-6】

对 ucfirst()函数的实例进行修改，要求将每个单词的首字母都变成大写。使用 ucwords()函数的实现代码如下所示：

```
<?php
$str1="those customers you are looking for, are looking for you - on Google";
echo "使用 ucwords()函数之后为: ".ucwords($str1);
$str2="歌曲名称 the day you went away";
echo "\n 使用 ucwords()函数之后为: ".ucwords($str2);
?>
```

执行后的结果如下所示：

```
使用 ucwords()函数之后为:  Those Customers You Are Looking For, Are Looking For
You - On Google
使用 ucwords()函数之后为: 歌曲名称:  The Day You Went Away
```

由于 PHP 默认以空格为单位分隔每个单词。所以，在此示例中“歌曲名称：The Day You Went Away”将作为一个单词来处理，而它的第一个字符是汉字，因此 ucwords()函数没有对它进行处理。

6.3.2 去除空格和特殊字符

除了大小写替换外，有时候还需要把字符串中存在的空格和特殊字符去掉。这包括去除字符串左侧的指定字符、去除字符串右侧的指定字符和同时去除字符串两侧的指定字符。

1. 去除字符串左侧的指定字符

ltrim()函数用于去除字符串左侧的指定字符，语法格式如下：

```
string ltrim ( string $str [, string $charlist ] )
```

默认删除字符串$str 左侧的特殊字符，这些字符包括：空格符、水平制表符（\t）、换行符（\n）、回车符（\r）、空值（\0）和垂直制表符（\x0b）。可选参数$charlist 用于指定要删除的其他字符。

【实践案例 6-7】

例如，下面的代码演示了如何使用 ltrim()函数：

```
<?php
$str1="    短消息(5 条)";                               //第 1 个字符串
echo "\$str1 使用 ltrim()之后结果: \"".ltrim($str1)."\"";  //去除左侧的空格
$str2="///////字符串处理///////";                    //第 2 个字符串
echo "\n\$str2 使用 ltrim()之后结果: \"".ltrim($str2)."\"";
                                                     //去除左侧的空格
echo "\nltrim(\$str2,\"/\")之后结果: \"".ltrim($str2,"/")."\"";
                                                     //去除左侧的字符/
$str3="\t\n 最新公告";                               //第 3 个字符串
echo "\n\$str3 使用 ltrim()之后结果: \"".ltrim($str3)."\"";
                                                     //去除左侧的水平制表符和换行符
?>
```

在查看输出结果之前，首先分析上面代码创建的字符串。在第 1 个字符串$str1 中使用了空格来填充；第 2 个字符串$str2 使用了自定义的符号"/"；第 3 个字符串$str3 则在左面使用了水平制表符（\t）和换行符（\n）。

执行之后的输出结果如下：

```
$str1 使用 ltrim()之后结果: "短消息(5 条)"
$str2 使用 ltrim()之后结果: "///////字符串处理///////"
ltrim($str2,"/")之后结果: "字符串处理///////"
$str3 使用 ltrim()之后结果: "最新公告"
```

2. 去除字符串右侧的指定字符

rtrim()函数的作用与 ltrim()函数相反，用于去除字符串右侧指定字符，语法格式如下所示：

```
string rtrim ( string $str [, string $charlist ] )
```

各个参数的含义与 ltrim()函数相同，这里不再重复。下面的代码演示了如何使用 ltrim()函数：

```
<?php
$str1="序列号将在 3 天后过期            ";          //第 1 个字符串
echo "\$str1 使用 rtrim()之后结果: \"".rtrim($str1)."\"";  //去除右侧的空格
$str2="<a><b><c><d>=====\t\t\n";                 //第 2 个字符串
$result=rtrim($str2);                            //去除右侧的水平制表符和换行符
echo "\nrtrim(\$str2)之后结果: \"".$result."\"";
echo "\nrtrim(\$result,\"=\")之后结果: \"".rtrim($result,"=")."\"";
                                                 //去除右侧的字符=
$str3="today is Friday";
echo "\nrtrim(\$str2,\"day\")之后结果: \"".rtrim($str3,"day")."\"";
                                                 //去除右侧的字符 day
?>
```

执行结果如下所示：

```
$str1 使用 rtrim()之后结果: "序列号将在 3 天后过期"
rtrim($str2)之后结果: "<a><b><c><d>====="
rtrim($result,"=")之后结果: "<a><b><c><d>"
rtrim($str2,"day")之后结果: "today is Fri"
```

3. 同时去除字符串两侧的指定字符

trim()函数实现了 ltrim()函数与 rtrim()函数的结合效果，可以同时去除字符串两侧的指定字符。语法格式如下所示：

```
string trim ( string $str [, string $charlist ] )
```

它默认删除字符串$str 两侧的一些字符，同样可以在可选参数$charlist 中指定要删除的其他字符。下面的代码演示了如何使用 trim()函数：

```
<?php
$str1="                    使用说明书                      ";
echo "\$str1 使用 trim()之后结果: \"".trim($str1)."\"";
$str2="   ///////字符串处理///////    ";
$result=trim($str2);
echo "\ntrim(\$str2)之后结果: \"".$result."\"";
echo "\ntrim(\$result,\"/\")之后结果: \"".trim($result,"/")."\"";
$str3="<><>最新公告<><><>";
echo "\ntrim(\$str3,\"<>\")之后结果: \"".trim($str3,"<>")."\"";
?>
```

执行结果如下所示：

```
$str1 使用 trim()之后结果："使用说明书"
trim($str2)之后结果："///////字符串处理///////"
trim($result,"/")之后结果："字符串处理"
trim($str3,"<>")之后结果："最新公告"
```

6.3.3 比较字符串

PHP 提供了 4 个函数进行字符串的比较，分别为 strcasecmp()、strcmp()、strncasecmp()和 strncmp()。

1. strcasecmp()函数

strcasecmp()函数可以执行不区分大小写的比较，语法格式如下：

```
int strcasecmp ( string $str1 , string $str2 )
```

如果$str1 小于$str2，则返回小于 0 的值；如果$str1 大于$str2，则返回大于 0 的值；如果$str1 等于$str2，则返回 0。

【实践案例 6-8】

根据 strcasecmp()函数的返回值编写一个比较两个字符串大小的功能，具体代码如下所示：

```
<?php
function CompareString($str1, $str2)
{
    $result=strcasecmp($str1, $str2);   //比较$str1 和$st2，将结果保存到
$result
    if($result==0)
    {
        echo "两个字符串相同";
    }elseif ($result>0)
    {
        echo "$str1 大于 $str2";
    }
    else {
        echo "$str1 小于 $str2";
    }
}
CompareString("hello","hello");         //输出：两个字符串相同
CompareString("PHP","php");             //输出：两个字符串相同
CompareString("php","php5");            //输出：php 小于 php5
CompareString("Good9","Good");          //输出：Good9 大于 Good
?>
```

2. strcmp()函数

strcmp()函数实现的功能与 strcasecmp()基本相同，不同的是它在比较时区分大小写。语法格式如下：

```
int strcmp ( string $str1 , string $str2 )
```

下面的代码演示如何使用 strcmp()函数进行比较：

```
<?php
if(strcmp("Hello","hello")==0)  //比较 Hello 和 hello
    echo "两个字符串相同";
else
    echo "两个字符串不相同";        //由于 strcmp()函数区分大小写，所以输出此行
?>
```

3. strncasecmp()函数

strncasecmp()函数与 strcasecmp()的功能相同，在比较时不区分大小写。不同之处在于使用 strncasecmp()可以指定两个字符串比较时使用的长度（即最大比较长度）。语法格式如下：

```
int strncasecmp ( string $str1 , string $str2 , int $len )
```

下面的代码演示如何使用 strncasecmp()函数进行比较：

```
<?php
$word1="FRIDAY";
$word2="Friend";
if(strncasecmp($word1,$word2,3)==0)     //比较$str1 和$str2 的前 3 个字符
    echo "两个字符串相同";               //输出此行
else
    echo "两个字符串不相同";
if(strncasecmp($word1,$word2,6)==0)     //比较$str1 和$str2 的前 6 个字符
    echo "两个字符串相同";
else
    echo "两个字符串不相同";             //输出此行
?>
```

4. strncmp()函数

strncmp()函数与 strcmp()的功能相同，在比较时区分大小写。不同之处在于使用 strncmp()可以指定两个字符串比较时使用的长度（即最大比较长度）。语法格式如下：

```
int strncmp ( string $str1 , string $str2 , int $len )
```

例如，下面的代码演示如何使用 strncmp()函数进行比较：

```
<?php
$word1="SEArch";
$word2="season";
if(strncmp($word1,$word2,3)==0) //比较$str1 和$str2 的前 3 个字符
    echo "两个字符串相同";
else
    echo "两个字符串不相同";       //由于 strncmp()函数区分大小写，所以输出此行
?>
```

6.3.4　查找字符串

查找（检索）字符串是指在一个字符串中查找某个子串的位置，包括第一次出现的位置、最后一次出现的位置，以及出现的总次数等。

1．stripos()函数

stripos()函数可以在字符串$str1 中查找$str2 第一次出现的位置，语法格式如下：

```
int stripos ( string $str1 , string $str2 [, int $offset = 0 ] )
```

该函数在查找时不区分大小写，$offset 可选参数用于指定查找的起始位置。如果找到则返回该位置，否则返回 false。

下面的代码演示如何使用 stripos()函数查找字符串。

```
<?php
$words="my heart will go on";
echo "查找 M 的出现位置：".stripos($words,"M");
echo "\n 查找 ea 的出现位置：".stripos($words,"ea");
echo "\n 查找 GO 的出现位置：".stripos($words,"GO");
echo "\n 从第 4 个开始查找 M 的出现位置：".stripos($words,"M",4);
?>
```

执行结果如下所示：

```
查找 M 的出现位置：0
查找 ea 的出现位置：4
查找 GO 的出现位置：14
从第 4 个开始查找 M 的出现位置:
```

2．strrpos()函数

strrpos()函数的功能与 stripos()函数相反，它可以在字符串$str1 中查找$str2 最后一次出现的位置。该函数语法格式如下：

```
int strrpos (string $str1 , string $str2 [, int $offset = 0 ] )
```

strops()函数在查找时区分大小写，其参数与 stripos()函数相同。下面的代码演示如何

使用 strrpos()函数查找字符串：

```
<?php
$words="food sheet would get";
echo "查找 d 最后的出现位置：".strrpos($words,"d");
echo "\n 查找 D 最后的出现位置：".strrpos($words,"D");
echo "\n 查找 et 最后的出现位置：".strrpos($words,"et");
echo "\n 从第 10 个开始查找 o 最后的出现位置：".strrpos($words,"o",10);
?>
```

执行结果如下所示：

```
查找 d 最后的出现位置：15
查找 D 最后的出现位置：
查找 et 最后的出现位置：18
从第 10 个开始查找 o 最后的出现位置：12
```

3. strripos()函数

stripos()函数的功能与 strrpos()函数基本相同，不同的是它在查找时忽略大小写。语法格式如下：

```
int strripos (string $str1 , string $str2 [, int $offset = 0 ] )
```

例如，下面的代码演示如何使用 strripos()函数查找字符串：

```
<?php
$words="food sheet would get";
echo "查找 ET 最后的出现位置：".strripos($words,"ET");
echo "\n 查找 D 最后的出现位置：".strripos($words,"D");
echo "\n 从第 4 个开始查找 OD 最后的出现位置：".strripos($words,"OD",4);
?>
```

执行结果如下所示：

```
查找 ET 最后的出现位置：18
查找 D 最后的出现位置：15
从第 4 个开始查找 OD 最后的出现位置：
```

4. substr_count()函数

substr_count()函数可以在字符串$str1 中查找$str2 出现的总次数。该函数语法格式如下：

```
int substr count (string $str1 , string $str2 [, int $offset = 0 [, int
$length ]] )
```

$offset 可选参数用于指定查找的开始位置，$lenght 可选参数用于指定最大搜索长度。

如果$offset 加上$length 的和大于$str1 的总长度，则打印警告信息。执行后该函数返回整型，表示出现的次数。

下面的代码演示如何使用 substr_count()函数查找字符串：

```
<?php
$text = 'This is a test';
echo strlen($text);                    //输出：14
echo substr_count($text, 'is');        //输出：2
echo substr_count($text, 'is', 3);  //字符串被简化为 's is a test'，因此输出 1
echo substr_count($text, 'is', 3, 3);   //字符串被简化为 's i'，所以输出 0
echo substr_count($text, 'is', 5, 10);  //因为 5+10 > 14，所以生成警告
$text2 = 'abcdabcda';
echo substr_count($text2, 'abcda');     //输出 1，因为该函数不计算重叠字符串
?>
```

6.4 操作子字符串

关于子字符串的操作大致可分为字符串的分隔、填充、截取和替换四大类，本节将一一介绍与之相关的函数。

6.4.1 分隔字符串

分隔字符串是指按照指定的分隔符，将一个字符串分隔成若干个子串。例如，将字符串“春|夏|秋|冬”按照分隔符“|”可以分隔成“春”、“夏”、“秋”和“冬”这 4 个字符串。

PHP 提供了 3 个函数进行分隔字符串，分别为 strtok()、explode()和 implode()。

1. strtok()函数

strtok()函数可以按指定的字符分隔字符串，它的语法格式如下：

```
string strtok ( string $str , string $token )
```

其中，$str 为要分隔的原始字符串，$token 为由分隔字符组成的字符串。strtok()函数在使用时比较特殊，在连续使用的时候，第一次需要指定字符串和分隔符，但是第二次只要放入分隔符就可以。一次可以只指定一个分隔符，也可以指定多个分隔符。

例如，下面的代码演示如何使用 strtok()函数分隔字符串：

```
<?php
$str="100|QQ群|itzcn.com";
$token = "|";
$result = strtok($str, $token);          //按“|”进行分隔，将结果保存到$result
echo "使用 strtok()后的结果：";
while($result !== false){
```

```
    echo "$result 、";
    $result = strtok($token);
}
$str="A B C,北京,上海;9 路;4 路;K2 路";
$token = " ,;";                      //指定 3 个分隔符
$result = strtok($str, $token);
echo "\n 使用 strtok()后的结果: ";
while($result!== false){
    echo "$result 、";
    $result = strtok($token);
}
?>
```

执行结果如下所示：

```
使用 strtok()后的结果: 100 、QQ 群 、itzcn.com 、
使用 strtok()后的结果: A 、B 、C 、北京 、上海 、9 路 、4 路 、K2 路 、
```

2. explode()函数

explode()函数可以根据指定的分隔符，将字符串分隔成一个由字符串组成的数组，数组中的每个元素都是字符串的一个子串。它的语法格式如下：

```
array explode ( string $separator , string $string [, int $limit ] )
```

$limit 参数用于限制分隔后的元素个数。如果设置了$limit 参数，则返回的数组包含最多$limit 个元素，而最后那个元素将包含$separator 的剩余部分。如果分隔符为空字符串（""），该函数将返回 false。如果分隔符所包含的值在$separator 中找不到，那么该函数将返回包含$separator 的单个元素的数组。如果$limit 参数是负数，则返回除了最后的-$limit 个元素外的所有元素。

例如，下面的代码演示如何使用 explode()函数分隔一个字符串：

```
<?php
$keys= "PHP ASP JSP ASP.NET";
$token = " ";
$result = explode($token,$keys);    //按$token 进行分隔，将结果保存到$result
print_r($result);                   //第 1 个数组
print_r(explode($token,$keys,2));   //第 2 个数组
print_r(explode($token,$keys,-2));  //第 3 个数组
?>
```

执行后将输出 3 个数组，结果如下所示：

```
Array
(
    [0] => PHP
    [1] => ASP
```

```
    [2] => JSP
    [3] => ASP.NET
)
Array
(
    [0] => PHP
    [1] => ASP JSP ASP.NET
)
Array
(
    [0] => PHP
    [1] => ASP
)
```

3. implode()函数

implode()函数的功能与 explode()函数相反。使用它可以指定一个分隔符，将一个字符串数组中的元素连接起来，组成一个字符串。语法格式如下：

```
string implode ( string $glue , array $pieces )
```

例如，下面的代码演示如何使用 implode()函数连接一个字符串：

```
<?php
$ary=array("苹果","西瓜","香蕉","葡萄","桔子");      //创建一个字符串数组
$fruits=implode("-",$ary);                         //将各个元素之间使用"-"分隔
echo $fruits;
?>
```

执行结果如下所示：

```
苹果-西瓜-香蕉-葡萄-桔子
```

6.4.2 填充字符串

填充字符串是指向字符串添加指定的字符，需要用到 PHP 的 str_pad()函数，它的语法格式如下：

```
string str_pad ( string $input , int $pad_length [, string $pad_string =
" " [, int $pad_type = STR_PAD_RIGHT ]] )
```

其中，$input 表示待填充的字符串，$pad_length 表示填充后的字符串长度，默认在字符串的右侧使用空格进行填充。该函数还有两个可选参数：$pad_string 用于指定填充字符；$pad_type 用于指定填充样式，它有如下可选值。

- **STR_PAD_RIGHT　在字符串右侧进行填充，此值为默认值。**
- **STR_PAD_LEFT　在字符串左侧进行填充。**

- **STR_PAD_BOTH　在字符串的两侧同时进行填充，如果无法对称，则优先填充右侧。**

【实践案例 6-9】

编写一个程序，要求实现能够对一个字符串的左侧、右侧和两侧插入指定字符串的功能。

根据题意使用 str_pad()函数是最合适的，如下为实例的测试代码：

```
<?php
$str1="真的好想你";
var_dump(str_pad($str1,30));
var_dump(str_pad($str1,30,"\\"));
var_dump(str_pad($str1,30,"*",STR_PAD_LEFT));
var_dump(str_pad($str1,30,"$",STR_PAD_BOTH));
$str2="十五的月亮";
var_dump(str_pad($str2,20,">"));
var_dump(str_pad($str2,21,"<>",STR_PAD_BOTH));
$str3="tell me why";
var_dump(str_pad($str3,21,"__",STR_PAD_BOTH));
?>
```

执行结果如下所示：

```
string(30) "真的好想你               "
string(30) "真的好想你\\\\\\\\\\\\\\\"
string(30) "***************真的好想你"
string(30) "$$$$$$$真的好想你$$$$$$$$"
string(20) "十五的月亮>>>>>"
string(21) "<><十五的月亮<><"
string(21) "_____tell me why_____"
```



在使用此函数时，如果指定的$pad_length 小于等于原字符串的长度，则不会进行填充，并且不会影响原字符串的长度。

6.4.3　截取字符串

截取字符串是指从一个字符串中提取一部分子串。例如从“我的祖国”这个字符串中截取出“祖国”这个子串。

PHP 提供了 5 个函数进行截取字符串，分别为 strstr()、stristr()、strpos()、strrchr()和 substr()。

1. strstr()函数

strstr()函数返回$haystack 中从$needle 的第一次出现到最后的部分，如果没有查找到则

返回 false。语法格式如下：

```
string strstr( string $haystack , mixed $needle [, bool $before_needle =
false ] )
```

在使用时要注意 strstr()函数查找时区分大小写。例如，下面的示例演示了它的用法：

```
<?php
$strs= "梅花三弄/一弄断人肠/二弄费思量 /三弄风波起";
echo strstr($strs, "/");         //输出: /一弄断人肠/二弄费思量 /三弄风波起
$strs= "today is Wednesday";
echo strstr($strs, "day");       //输出: day is Wednesday
echo strstr($strs, "Day");       //区分大小写没有输出
$strs="zhht@126.com";
echo strstr($strs, "@");         //输出: @126.com
?>
```

2. stristr()函数

stristr()函数和 strstr()函数实现的功能相同，只是它在查找时不区分大小写。语法格式如下：

```
string stristr (string $str1 , mixed $str2 )
```

例如，下面的示例演示了它的用法：

```
<?php
$strs= "today is Wednesday";
echo stristr($strs, "DAY");              //输出: day is Wednesday
?>
```

3. strpos()函数

strpos()函数返回$haystack 中$needle 第一次出现的整数位置。语法格式如下：

```
int strpos ( string $haystack , mixed $needle [, int $offset = 0 ] )
```

该函数查找时区分大小写。如果找到返回的整数位置指的是第多少个字节，从 0 开始计数；如果没有找到则返回 false。

下面的代码使用 strpos()函数从$str1 字符串查找并输出结果：

```
<?php
$strs= "today is Wednesday";
$i=strpos($strs, "day");
echo "day 第一次出现在\$strs 的 $i 位置。";
$strs= "today is Wednesday";
$i=strpos($strs, "DAY");
echo "day 第一次出现在\$strs 的 $i 位置。";
```

```
?>
```

执行结果如下所示：

```
day 第一次出现在$strs 的 2 位置。
day 第一次出现在$strs 的  位置。
```

4. strrchr()函数

strrchr()函数返回$str1 中从$str2 的最后一次出现到最后的部分；如果没有查找到则返回 false。语法格式如下：

```
string strrchr (string $str1 , mixed $str2 )
```

该函数在查找如果$str2 中包含了不止一个字符，那么仅使用第一个字符。下面的代码使用 strrchr()函数从$str1 字符串查找并输出结果：

```
<?php
$strs= "today is Wednesday";
echo strrchr($strs, "Wen");          //输出: Wednesday
echo strrchr($strs, "WABC");         //输出: Wednesday
?>
```

5. substr()函数

substr()函数返回$string 中从第$start + 1 个字节开始到最后的部分。语法格式如下：

```
string substr ( string $string , int $start [, int $length ] )
```

对于$start 参数它有如下 3 种情况。

- 非负数。返回的字符串将从$string 的$start 位置开始，从 0 开始计算。例如，在字符串“abcdef”中，在位置 0 的字符是“a”，位置 2 的字符串是“c”等。
- 是负数。返回的字符串将从$string 结尾处向前数第$start 个字符开始。
- 如果$string 的长度小于或等于 start，将返回 false。

$length 是一个可选参数，它有如下 4 种情况。

- 正数。返回的字符串将从$start 处开始最多包括$length 个字符（取决于$string 的长度）。
- 负数。那么$string 末尾处的许多字符将会被漏掉（若$start 是负数，则从字符串尾部算起）。如果$start 不在这段文本中，那么将返回一个空字符串。
- 为 0，false 或 NULL 的$length，那么将返回一个空字符串。
- 如果没有提供$length，返回的子字符串将从 start 位置开始直到字符串结尾。

下面的代码演示了使用 substr()函数对字符串求子串，并输出结果：

```
<?php
$strs="hello everyone";
echo substr($strs,0);                   //输出: hello everyone
```

```
echo substr($strs,6);                    //输出：everyone
echo substr($strs,-6);                   //输出：eryone
echo substr($strs,2,5);                  //输出：llo e
echo substr($strs,2,-2);                 //输出：llo everyo
?>
```

6.4.4 替换字符串

PHP 提供了 4 个函数进行替换字符串，分别为 str_replace()、str_ireplace()、substr_replace()和 strtr()。

1. str_replace()函数

str_replace()函数可以搜索$subject 中的子字符串$search。如果找到了子串，则使用$replace 替换$search，然后返回$subject 的值。语法格式如下：

```
mixed str replace ( mixed $search , mixed $replace , mixed $subject [, int
&$count ] )
```

该函数的$count 可选参数用于统计$search 在 subject 中的匹配次数。例如，下面的代码演示如何使用 str_replace()函数替换字符串：

```
<?php
$html="<head>{title}</head>";
$title="<title>百度音乐盒</title><meta content='text/html; charset=gb2312'
/>";
echo str_replace("{title}", $title, $html);
echo "\n";
$colors="Red Black White Green";
$key=array("a","b","e");
echo str_replace($key, "*", $colors);
echo "\n";
$str="one two three four five six seven eight nine ten";
echo str replace("e", "E", $str, $count);
echo "\n";
echo "字母 e 一共在\$str 中出现了 $count 次";
?>
```

执行结果如下所示：

```
<head><title>百度音乐盒</title><meta content='text/html; charset=gb2312'
/></head>
R*d Bl*ck Whit* Gr**n
onE two thrEE four fivE six sEvEn Eight ninE tEn
字母 e 一共在$str 中出现了 9 次
```

2. str_ireplace()函数

str_ireplace()函数的功能与 str_replace()函数基本相同，不同的是它在替换时不区分大小写。语法格式如下：

```
mixed str ireplace ( mixed $search , mixed $replace , mixed $subject [, int &$count ] )
```

例如，下面的代码演示如何使用 str_ireplace()函数替换字符串：

```
<?php
$str="one two three four five six seven eight nine ten";
//源字符串
echo str_ireplace("E", "*", $str, $count);
echo "字母 e 一共在\$str 中出现了 $count 次";
?>
```

执行结果如下所示：

```
on* two thr** four fiv* six s*v*n *ight nin* t*n
字母 e 一共在$str 中出现了 9 次
```

3. substr_replace()函数

substr_replace()函数可以使用$replace 替换$subject 中从第$start 个字符开始，长度为$len 的子串，并返回$subject 的值。语法格式如下：

```
mixed substr replace ( mixed $string , string $replacement , int $start [, int $length ] )
```

下面的代码演示如何使用 substr_replace()函数替换字符串：

```
<?php
$str= '<head>{title}</head>';
/* 使用 "<html></html>" 替换整个 字符串*/
echo substr replace($str, '<html></html>', 0) . "\n";
echo substr replace($str, '<html></html>', 0, strlen($str)) . "\n";

/* 将 "<html>" 插入到 $str 的开头处。*/
echo substr replace($str, '<html>', 0, 0) . "\n";

/* 使用 "<title>窗内网</title>" 替换 $var 中的"{title}"*/
echo substr_replace($str, '<title>窗内网</title>', 6, 7) . "\n";

/* 从 $str 中删除 "{title}"*/
echo substr_replace($str, '', 6, 7) . "\n";
?>
```

执行结果如下所示：

```
<html></html>
<html></html>
<html><head>{title}</head>
<head><title>窗内网</title></head>
<head></head>
```

4．strtr()函数

strtr()函数可以将字符串中的指定字符进行替换，它有两种语法格式。第一种格式如下：

```
string strtr ( string $str, string $from, string $to )
```

这里共有 3 个参数，替换时会将第二个参数的字符串替换为第三个参数的字符串，如果两个字符串参数的长度不一致，则较长的那个字符串将会被截断。

第二种格式有两个参数，第二个参数是一个准备进行替换处理的数组。

```
string strtr ( string $str, array $replace_pairs )
```

例如，下面的代码演示如何使用 strtr()函数替换字符串：

```
<?php
$str = array('bj'=>"beijing",'web'=>"itzcn.com");
echo strtr("hello web in bj.", $str);
?>
```

执行后的输出结果如下：

```
hello itzcn.com in beijing.
```

6.5　习题

一、填空题

（1）在使用单引号定义的字符串中，如果要输出一个单引号需要使用代码__________。

（2）使用定界标识符创建字符串时，名称不能用__________和下划线作为开头。

（3）在下面空白填写合适的代码，使程序可以统计$str 字符串的长度。

```
$str="a b c d e f g";
$lenght=           ;
echo "字符串的长度是: ".$lenght;
```

（4）假设有下面一段 PHP 代码，执行后的输出为__________。

```
$result = strcmp("ABCD", "abcd");       //比较两个字符串
if($result === 0){
```

```
    echo "A";
}else{
    echo "B";
}
```

（5）假设有下面一段 PHP 代码，运行后$str 的结果为__________。

```
$str='http://www.itzcn.com/';
$str=substr($str,7);
```

二、选择题

（1）假设有一个变量$name 的值是“book”，下面关于该变量的使用不正确的是__________。

A．echo $name

B．echo $name[0]

C．echo $name{0}

D．echo $name"0"

（2）下列代码运行后不能创建一个字符串的是__________。

A．

```
$str='Let\'s say that it\'s true.';
```

B．

```
$str= "用这种方式也可以哦。
北京、上海、广州、重庆
";
```

C．

```
$str="创建一个名为\$width 的变量";
```

D．

```
$str=<<<58com
58.com
58com;
```

（3）strtolower()函数的作用是__________。

A．将字符串中的所有字符都转换成大写

B．将字符串中的所有字符都转换成小写

C．将字符串的第一个字符转换成大写

D．将字符串中的所有单词的第一个字符都转换成大写

（4）在下面空白处填写合适的代码，使程序的出结果是“2012 年下学期课程安排”。

```
$str = "  ---2012 年下学期课程安排---  ";
echo __________;
```

A．trim($str," -")

B．ltrim($str," -")

C．rtrim($str," -")

D．str_replace($str," -")

（5）假设有下面一段 PHP 代码，运行后$result 的结果为__________。

```
<?php
$s = "PHP";
$length = 20;
$result = str_pad($s, $length,"*");
?>
```

A．*P*H*P**************

B．********PHP*********

C．PHP*****************

D．*****************PHP

三、上机练习

1．对字符串的各种操作

使用本章学习的知识定义一个字符串，它的内容如下：

```
<script>
wpo.tti=new Date*1;baidu.g("kw")&&F.call('page/analyse', 'runWpoStat',
wpo);
</script>
```

然后编写程序实现如下操作。

- 提到 new 后面的字符串并输出。
- 将 baidu 替换为 itzcn.com 并输出。
- 统计字符 s 出现的次数和单词数量。
- 输出字符“(”第一次出现的位置。
- 将首尾的<script>和</script>去掉并输出。
- 将字符串从 10 往后的 5 个字符追加到尾部。

6.6 实践疑难解答

6.6.1 PHP 加法运算中如果包含了字符串是怎么处理的



PHP 加法运算中如果包含了字符串是怎么处理的

网络课堂：http://bbs.itzcn.com/thread-19687-1-1.html

【问题描述】：刚接触 PHP，对字符串的运算有点疑问。具体内容如下：

```
<?php
$a = "123abc";
echo "$a = " + $a ; //246
echo "<hr>";

$b = "abc123";
echo "$b = " + $b; //0
?>
```

对于上面的代码，还望各位帮忙解答一下，主要是不太清楚这两个的结果是怎么来的。似乎以数字开头就是数字，以其他内容开头是字符串……比较困惑，还望各位知道的朋友指教一二，谢谢。

【解决办法】：PHP 中对于字符串转数字是这样解释的。

当一个字符串被当作数字来求值时，根据以下规则来决定结果的类型和值。如果包括“.”、“e”或“E”其中任何一个字符，字符串被当作 float 来求值；否则就被当作整数，该值由字符串最前面的部分决定。如果字符串以合法的数字数据开始，就用该数字作为其值，否则其值为 0（零）。合法数字数据由可选的正负号开始，后面跟着一个或多个数字（可选的包括十进制分数），后面跟着可选的指数。指数是一个“e”或者“E”后面跟着一个或多个数字。

例如，下面的测试代码有助于你的理解：

```
<?php
$foo = 1 + "10.5";           // $foo is float (11.5)
$foo = 1 + "-1.3e3";              // $foo is float (-1299)
$foo = 1 + "bob-1.3e3";           // $foo is integer (1)
$foo = 1 + "bob3";                // $foo is integer (1)
$foo = 1 + "10 Small Pigs";       // $foo is integer (11)
$foo = 4 + "10.2 Little Piggies";   // $foo is float (14.2)
$foo = "10.0 pigs " + 1;          // $foo is float (11)
$foo = "10.0 pigs " + 1.0;        // $foo is float (11)
?>
```

6.6.2　提取 URL 中字符串参数的问题



提取 URL 中字符串参数的问题

网络课堂：http://bbs.itzcn.com/thread-19688-1-1.html

【问题描述】：假设有如下代码：

```
<?php
   $url=parse_url('http://localhost/test.php?x=4&y=5&z[]=6&z[]=7');
   parse_str($url['query'],$out);
```

```
    print r($out)
    //希望得到的输出: Array ( [x] => 4 [y] => 5 [z] => Array ( [0] => 6 [1]
    => 7 ) )
?>
```

问题是如何提取字符串后面的参数值，并放到数组中。我记得有个函数是专门处理这个的，但是试了几个函数都不行，为什么？

【解决办法】：如你所言，PHP 确实提供了分隔字符串的函数。下面是测试代码：

```
<?php
$paras=explode('&',$_SERVER["QUERY_STRING"]);
$arr=array();
for($i=0;$i<count($paras);$i++){
$rs=explode("=",$paras[$i]);
$arr[$rs[0]]=$rs[1];
}
print r($arr);
?>
```

假设运行的 URL 是 http://localhost/test.php?id=1&x=2&y=3，那么返回结果如下：

```
Array([id]=>1 [x]=>2 [y]=>3)
```

第 7 章 常用数据处理

前面介绍对数组和字符串的操作时，都是使用的 PHP 系统函数。PHP 自带了大量的系统函数，使用这些函数可以完成大部分的任务。同时，PHP 还允许用户编写自定义的函数实现系统函数无法实现的功能。

本章首先向读者介绍如何编写以及调用用户自定义函数。然后讲解 PHP 系统函数在四个方面的应用，分别是：数学运算、处理日期和时间、处理 XML 以及正则表达式。在结束本章的学习后，读者将对 PHP 有更深的了解，从而可以轻松构建出各种类型的网站。

本章学习要点：

- 掌握如何定义函数
- 掌握自定义函数的调用和传参方式
- 熟悉常用的数学运算函数
- 掌握获取时间和格式化日期的方法
- 熟悉 XML 文档的结构
- 掌握 DOM 读取和生成 XML 的方法
- 掌握 SimpleXML 读取和生成 XML 的方法
- 掌握 POSIX 正则表达式的定义和处理方式
- 掌握 Perl 正则表达式的定义和处理方式

7.1 用户函数

函数是完成一个特定功能的代码集合。按照类型可以分为系统函数和用户函数两种。例如，在前面两章介绍对数组和字符串的操作时就大量使用了系统函数。

系统函数由 PHP 提供，只能用于解决某些特定问题，且无法根据实际需求进行调整。这时，用户可以通过创建自定义函数来解决问题。

7.1.1 函数定义语法结构

在 PHP 中允许用户使用 function 关键字创建一个自定义函数，语法格式如下：

```
function 函数名称(参数 1, 参数 2, …)
{
    //函数内的代码
```

```
}
```

PHP 对函数名称的限制比较少，函数名称可以是以字母或下划线开头后跟字母、下划线或数字的任何字符串，而且不区分大小写，因此 Add 和 add 是指同一个函数。在括号内是函数的参数，多个之间用逗号分隔，没有参数时括号也不能省略。最后大括号内是函数体，在函数体内使用 return 语句可以指定函数的返回值。

【实践案例 7-1】

在创建一个函数之前，首先应该为它指定一个有意义的名称，该名称应能很好地提示函数的功能。例如，开发一个论坛系统，有一个名为 forum 的模块包含版块管理的函数，其中有一个查看版块是否隐藏的函数，可以使用 forum_is_hidden 或者 forumIsHidden 作为名称，当然也可以是 is_forum_hidden。

根据上面介绍的函数定义语法，创建一个名为 Sum()的函数，它实现计算两个数的之和的功能，并将结果返回，最终代码如下所示：

```
<?php
function Sum($number1,$number2)     //Sum()函数需要两个参数
{
    $Result=$number1+$number2;
    return $Result;                 //使用 return 指定返回值为两数之和
}
?>
```

在上述代码中，Sum()函数有两个参数$number1 和$number2，将这两个参数相加后保存到$Result 变量中。最后使用 return 语句将$Result 的值返回，即两数之和。

【实践案例 7-2】

在传统的 HTML 中，如果需要让字体显示为带下划线的粗斜体，那么至少需要用到 U 标记、B 标记和 I 标记。当然，如果网站中只有一处需要这样显示，可以直接使用这些标记。但是，如果在网站中有很多地方需要以这种方式显示，最简单的方法就是定义一个函数来实现输出。

下面是实现上述功能的函数定义：

```
<?php
function format_Html($text)             //带有一个参数的 format_Html()函数
{
    $text = "<u><i><b>$text</b></i></u>";       //应用加粗、加斜和粗体标记
    echo $text ;                        //输出格式化后的字符串
}
?>
```

7.1.2 使用函数

用户函数创建之后，便可以像系统函数一样使用，即通过指定函数名称来调用，如果该函数需要参数，需要在小括号内指定参数的值，但是必须注意参数的类型应与定义时

一致。

【实践案例 7-3】

创建一个 PHP 页面，编写程序使用 7.1.1 节定义的 Sum()函数和 format_Html()函数。

（1）创建一个 PHP 页面，并添加 7.1.1 节 Sum()函数和 format_Html()函数的定义代码。

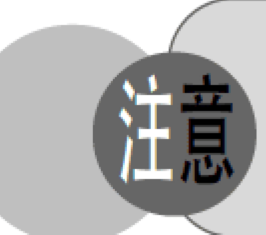
（2）在页面的合适位置添加如下函数的调用代码：

```
<TABLE cellSpacing=0 cellPadding=0 width="511" border=0>
   <TR>
     <TD width="149" rowSpan=5><DIV align=left><img border="0" src=
     "images/cp.jpg" width="127" height="165"></DIV></TD>
     <TD width="70" height=22>产品名称：</TD>
     <TD width="292" height=22 class="title"><?php format_Html("易胜通
     用企业管理系统"); ?></TD>
   </TR>
   <TR>
     <TD height=44 width="70">功    能：</TD>
     <TD height=44 width="292">1.信息管理 2.<?php format Html("简介管理
     "); ?> 3.用户管理 4.新闻管理<br>      <br>   5.<?php format_Html("产
     品管理"); ?> 6.留言管理 7.联系方式管理 8.连接管理</TD>
   </TR>
   <TR>
     <TD height=22 width="70">销    量：</TD>
     <TD height=22 width="292"><?php echo Sum(100,300);?></TD>
   </TR>
</TABLE>
```

（3）浏览 PHP 页面查看运行效果，如图 7-1 所示。



图 7-1　调用自定义函数运行效果



在调用用户自定义函数时，必须保证在调用之前函数已经存在，即函数应该先定义再调用，否则将无法运行。

7.1.3 函数返回值

有时需要在程序外部使用函数执行的结果，这时就需要在函数内使用 return 语句指定一个返回值。例如，在 7.1.2 节的 Sum()函数中使用 return 语句返回求和的结果，该语句只能用于函数内。在函数内可以有多个 return 语句，但是同时只能有一个语句被执行。

【实践案例 7-4】

例如，编写一个函数返回两个数中的较大数，代码如下：

```
<?php
function MaxNumber($number1,$number2)   //求最大数函数
{
    if($number1>$number2)
        return $number1;                //返回第一个参数
    else
        return $number2;                //返回第二个参数
}
echo "MaxNumber(5,6)的返回值是: ".MaxNumber(5,6);
                                        //输出“MaxNumber(5,6)的返回值是: 6”
echo "MaxNumber(8,7)的返回值是: ".MaxNumber(8,7);
                                        //输出“MaxNumber(8,7)的返回值是: 8”
?>
```

【实践案例 7-5】

使用 return 语句可以为函数返回任何类型的数据。例如，下面代码演示如何使用函数返回数组并遍历输出：

```
<?php
function getDataAry()
{
    $resAry=array(95,87,79,80,62,74,90,92);     //创建一个数组
    return $resAry;                         //返回该数组
}
$ary=getDataAry();                          //保存 getDataAry()函数返回的数组
foreach ($ary as $i)                        //遍历数组
    echo $i.",";                            //输出所有数
?>
```

7.1.4 函数参数传递方式

PHP 支持的参数传递方式有：按值传递、按引用传递、默认值传递和可变参数列表传递，下面详细介绍每种方式。

1. 按值传递

按值传递是 PHP 默认的参数传递方式，这种传递方式将为函数外部变量的值创建一个副本，然后赋给函数内部的局部变量。在函数处理完成后，该外部变量的值不发生改变，除非在函数内部声明该外部变量作用域为全局。

【实践案例 7-6】

下面创建一个 PassByValue()函数演示按值传递前后参数值的变化。

```
<?php
function PassByValue($number,$str)        //按值传递参数
{
    $number+=100;                          //第一个参数增加 100
    $str.=" World";                        //第二个参数附加 World 字符串
    echo "函数内\$number=", $number, ", \$str=",$str,"<br>";
                                           //输出两个参数的结果
}
$number=3;                                 //创建一个整数作为第一个参数
$str="hello";                              //创建一个字符串作为第二个参数
PassByValue($number,$str);                 //调用 PassByValue()函数
echo "函数外\$number=", $number, ", \$str=",$str,"<br>";
                                           //输出调用两个参数的值
?>
```

在上述代码中，PassByValue()函数的外部定义了两个变量，分别为$number 和$str，在 PassByValue()函数的内部，对这两个变量重新进行了赋值，然后输出这两个变量的值。另外，在调用 PassByValue()函数后，再次将这两个变量进行输出，从而来观察这些变量的值有没有发生变化。

输出结果如下所示：

```
函数内$number=103, $str=hello World
函数外$number=3, $str=hello
```

从上述结果中可以看出，在 PassByValue()函数中这两个变量的值发生了变化，但是在调用函数后，这两个变量又恢复为它们的初始值。



PHP 的按值传递参数，在函数范围内对这些值的任何改变在函数外部都会被忽略。

2. 按引用传递

在按引用传递参数方式下，实参的内存地址被传递到形参中，在函数内部对形参的任何修改都会影响到实参，因为它们被存储到同一个内存地址。函数返回后，实参的值将会发生变化。引用传递参数的形参和实参都是针对同一个块地址修改的。如果希望一个函数

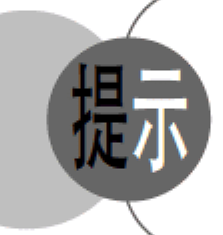
参数通过引用被传递，需要在函数定义的参数名前面添加符号&来实现。

例如，将按值传递参数的示例修改为按引用方式传递参数，只需要在 PassByValue()函数的两个参数前面分别添加符号&。修改后的函数如下：

```
function PassByValue(&$number,&$str)        //按引用传递参数
{
    $number+=100;                           //第一个参数增加 100
    $str.=" World";                         //第二个参数附加 World 字符串
    echo "函数内\$number=", $number, ", \$str=",$str,"<br>";
                                            //输出两个参数的结果
}
```

再次运行将看到如下的输出，从中可以看出，在函数内两个变量的值发生了变化，并且这种变化的结果将影响到函数调用结束后。

```
函数内$number=103, $str=hello World
函数外$number=103, $str=hello World
```



使用通过引用传递参数的方式时，在函数内对这些值的任何改变，在函数之外也能反映出这些修改。

3. 默认值传递

除了按值传递参数和按引用传递参数的方式外，一个函数还可以使用预先定义好的默认参数。在未指定参数的情况下，函数使用默认值作为函数的参数；在提供了参数的情况下，函数使用指定的参数。

【实践案例 7-7】

例如，下面创建一个用于为指定文本设置字体颜色的 setFontColor()函数。该函数有两个参数，第一个参数指定文本，第二个参数指定字体颜色，其中的第二个参数具有默认值，代码如下所示：

```
<?php
function setFontColor($str,$color="red")    //创建带默认值的参数
{
    echo "<font color='".$color."'>".$str."</font><br/>";
}

setFontColor("教程");                       //使用参数的默认值
setFontColor("热门商品","black");           //修改参数的默认值
?>
```

在上述代码中，调用 setFontColor()函数时，可以传递两个参数也可以传递一个参数。如果只传递一个参数，则第二个参数使用创建函数时定义的默认值。执行后的输出结果如

下所示：

```
<font color='red'>教程</font><br/>
<font color='black'>热门商品</font><br/>
```

在使用 PHP 的默认参数时需要注意，默认值必须是常量表达式，不能是变量。如果函数有多个参数，可以为多个参数指定默认值。但是，带默认值的参数只能位于参数列表的最后，即一个默认值参数的右边不能出现没有指定默认值的参数。

例如，如下面的示例就是错误的：

```
function setPoint($x,$y=0,$z)        //错误
{
    echo "{",$x,",",$y,",",$z,"}";
}
setPoint (5,7);
```

在上述代码中，使用 setPoint(5,7)调用 setPoint()函数时，将会出现二义性错误。因为无法确定 7 应该传递给$y 还是$z。可以将上述代码调整为如下正确形式：

```
function setPoint($x,$z,$y=0)        //正确：将带默认值的参数放在最后
{
    echo "{",$x,",",$y,",",$z,"}";
}
```

4. 可变参数列表传递

在 PHP 中还有一种特殊的参数传递方式——可变参数列表，即参数的数量是不确定的。这种方式需要借助 3 个特殊的函数获取传入的参数。表 7-1 列出了它们及其说明。

表 7-1　可选参数函数

名称	格式	说明
func_num_args()	func_num_args(void)	返回自定义的函数中传入的参数个数
func_get_arg()	func_get_arg($arg_num)	取得第$arg_num+1 个参数的值
func_get_args()	func_get_args(void)	返回一个包含所有参数的数组

【实践案例 7-8】

创建一个函数可以实现对调用时传递的任意数量的数字进行排序，并以降序形式输出。

根据前面介绍的函数知识，由于无法确定参数的个数，使用可选参数列表是最适合的。这种方式并不需要特别的语法，参数列表仍按函数定义的方式传递给函数，并按通常的方式使用这些参数。

最终的实现代码如下所示：

```
<?php
function sortNumbers()                    //排序函数
{
```

```
    $count=func_num_args();              //获取实际传递的参数个数
    $ary=func_get_args();                //获取所有参数列表的数组
    rsort($ary);                         //对数组进行排序
    echo "本次排序的共有 $count 个数字，结果如下：\n";
    foreach ($ary as $n) {
        echo " $n";                      //输出排序后的数字
    }
    echo "\n";
}
sortNumbers(3,5,2,56,32,75,74,82,53,66,79,46); //对 12 个数字排序
sortNumbers(59,26,46,31,89,47);          //对 6 个数字排序
?>
```

上述代码运行后的输出如下：

```
本次排序的共有 12 个数字，结果如下：
 82 79 75 74 66 56 53 46 32 5 3 2
本次排序的共有 6 个数字，结果如下：
 89 59 47 46 31 26
```

在上面创建了一个 sortNumbers()函数，该函数在定义时没有参数。在函数内使用 func_num_args()函数获得实际调用时参数的数量并保存到$count 变量中。接下来使用 func_get_args()函数获得所有传递的参数，并以数组的形式保存到$ary 变量中，然后对$ary 进行降序排列，最后输出。

7.1.5 递归函数

递归函数是指在一个函数的函数体内调用函数本身。在递归函数中，主调函数又是被调函数，递归函数反复调用其自身，每调用一次进入新的一层。例如，要遍历一个目录的内容，目录中包含有多个子目录，子目录中又包含有下级目录。

【实践案例 7-9】

编写一个递归函数实现计算 1+2+3+…+N 的和的功能，要求 N 作为函数的参数。函数的代码如下所示：

```
<?php
function sum($number)                    //递归函数
{
    if($number!=0)                       //判断是否停止递归
    {
        return $number+sum($number-1);   //在返回值中调用本函数
    }
}
echo "100 求和结果：".sum(100);           //输出“100 求和结果：5050”
?>
```

从上述代码可以看到，递归函数只需少量的程序即可描述出解题过程所需要的多次重复计算，大大减少了程序的代码量。但是，要注意必须为递归函数设置停止条件，否则将造成死循环。

7.1.6 嵌套函数

嵌套函数是指在一个函数体中又同时定义一个函数，两个函数形成嵌套关系。此时只有外部函数被调用后，内部函数才能使用。

例如，下面的示例演示了嵌套函数的使用：

```
<?php
function start(){                           //外部函数
    echo "正在开机...\n";
    function boot() {                       //内部函数
        echo "正在加载引导程序...\n";
    }
    function welcome($user) {               //内部函数
        echo "欢迎[ $user ]使用本系统。\n";
    }
}
start();            //调用外部函数 start()，此时两个内部函数均变得可用
boot();
welcome("zhht");
?>
```

上述代码共定义了 3 个函数，start()是外部函数，其中包含了 boot()和 welcome()两个内部函数。因此，为了使用 boot()和 welcome()函数，必须先调用 start()函数，否则将提示函数未定义。运行后的输出如下：

```
正在开机...
正在加载引导程序...
欢迎[ zhht ]使用本系统。
```

7.1.7 判断函数是否存在

在开发大型项目时，通常都是多人协作的团队开发，这时候就要避免自定义函数名称重复的情况。在 PHP 中可以使用 function_exists()函数判断指定的用户函数是否已经存在。

例如，下面代码在调用 userLogin()函数之前首先判断该函数是否存在。如果已经存在则直接调用，否则先创建。

```
<?php
if(!function_exists("userLogin"))           //判断 userLogin()函数是否存在
{
    function userLogin($u)  {               //如果不存在则创建
```

```
        echo "用户 $u 登录成功";
    }
}
userLogin("zhht");                      //调用 userLogin()函数
?>
```

还可以使用 create_function()函数创建一个临时函数，这个函数名称由 PHP 动态生成，从而避免名称相同的情况。

例如，下面代码使用 create_function()函数创建动态函数实现前面 userLogin()函数的功能。

```
<?php
$userLogin=create_function('$u', 'echo "用户 $u 登录成功";');
echo $userLogin("zht");                //输出“用户 zht 登录成功”
?>
```

7.2 数学运算

数学运算是程序中最常执行的功能之一，为此 PHP 提供了很多系统函数实现运算功能。它们无须安装、编译和配置，便可以直接使用。但是要注意，这些数学函数仅能处理计算机中 integer 和 float 范围的值。表 7-2 中列出了常用的数学函数。

表 7-2 常用数学函数

函数名称	功能描述	函数名称	功能描述
abs()	绝对值	min()	最小值
asin()	反正弦	pow()	指数表达式
ceil()	进一法取整	rand()	产生一个随机数
decbin()	十进制转换为二进制	round()	对浮点数进行四舍五入
floor()	舍去法取整	sin()	平弦
max()	最大值	sqrt()	平方根

【实践案例 7-10】

根据表 7-2 给出的数学函数创建一个案例，演示它们的具体作用。

创建一个名为 Math.php 的 PHP 文件，添加具体代码如下所示：

```
<?php
echo "使用 abs(-2541.8)函数求绝对值: ".abs(-2541.8);
echo "<br/>使用 asin(0.6)函数求反正弦: ".asin(0.6);
echo "<br/>使用 ceil(9.91)函数求一个整数: ".ceil(9.91);
echo "<br/>使用 decbin(16)函数将 16 转换为二进制: ".decbin(16);
echo "<br/>使用 floor(9.91)函数求一个整数: ".floor(9.91);
echo "<br/>使用 max(9,100,20,83,-20)函数求最大值: ".max(9,100,20,83,-20);
echo "<br/>使用 min(9,100,20,83,-20)函数求最小值: ".min(9,100,20,83,-20);
echo "<br/>使用 pow(2,4)函数求指数: ".pow(2,4);
```

```
echo "<br/>使用 rand()函数产生一个随机数: ".rand();
echo "<br/>使用 round(9.991)函数求一个整数: ".round(9.991);
echo "<br/>使用 sin(60)函数求正弦: ".sin(60);
echo "<br/>使用 sqrt(81)函数求平方根: ".sqrt(81);
?>
```


运行后，将看到如下输出结果：

```
使用 abs(-2541.8)函数求绝对值: 2541.8
使用 asin(0.6)函数求反正弦: 0.64350110879328
使用 ceil(9.91)函数求一个整数: 10
使用 decbin(16)函数将 16 转换为二进制: 10000
使用 floor(9.91)函数求一个整数: 9
使用 max(9,100,20,83,-20)函数求最大值: 100
使用 min(9,100,20,83,-20)函数求最小值: -20
使用 pow(2,4)函数求指数: 16
使用 rand()函数产生一个随机数: 24939
使用 round(9.991)函数求一个整数: 10
使用 sin(60)函数求正弦: -0.30481062110222
使用 sqrt(81)函数求平方根: 9
```

在本示例中需要注意的是，round 函数、floor 函数和 ceil 函数都可以实现取一个浮点数的整数的功能，但是它们的取整规则不一样。

【实践案例 7-11】

在很多系统中使用密码找回功能时，系统会产生一个随机密码作为用户的新密码，这主要是使用了随机函数。下面创建一个函数用于产生指定长度的随机密码，默认为 6 位。

如下所示为最终的实现代码：

```
<?php
function CreatePassword($length=6)             //生成密码函数，默认为 6 位
{
    $dictionary="abcdefghijklmnopqrestuvwxyz0123456789";   //密码字典
    $maxChar=strlen($dictionary)-1;            //获取字典长度
    $password="";
    for($i=0;$i<$length;$i++)                  //随机生成每一位
    {
        $password.=$dictionary[rand(0, $maxChar)];
                                               //产生随机数，再从字典中取字母
    }
    return $password;                          //返回密码
}
echo "用户 1 的密码: ".CreatePassword();
echo "\n 用户 2 的密码: ".CreatePassword();
echo "\n 用户 3 的密码: ".CreatePassword(10);
?>
```

在 CreatePassword()函数体内首先创建了一个长字符串作为产生随机密码的字典。然后在循环时产生一个随机数，最终形成一个随机的密码字符串。

上述代码执行后的结果如下所示。

```
用户 1 的密码: sxwcdm
用户 2 的密码: 17jdwp
用户 3 的密码: yuyzyy9xi3
```

7.3 日期和时间运算

与数学函数相比，PHP 在日期和时间方面的运算提供了更丰富的函数。例如，可以获取服务器的当前日期和时间，还可以将日期或者时间转换为很多不同的格式输出。

7.3.1 UNIX 时间戳

时间戳（T imestamp）是源于 UNIX 系统的时间表示方法，是指从 1970 年 1 月 1 日（00:00:00 GMT）起到现在所经过的秒数，因此也称为 UNIX 时间戳。

在 PHP 中使用时间戳的最简单方式就是调用 time()函数，time()函数可以根据当前时间返回一个时间戳的表示法，示例如下：

```
echo "当前时间戳为: ".time();
```

7.3.2 日期函数

PHP 提供的日期函数中最常用的有：date()、gmdate()、getdate()和 checkdate()。

1. date()函数

date()函数用于格式化一个本地日期和时间，它的语法格式如下：

```
string date ( string $format [, int $timestamp ] )
```

返回将整数$timestamp 按照$format 给定格式而产生的字符串。其中，$timestamp 参数是可选的，如果没有给出时间戳，则使用本地当前时间，即 time()。

表 7-3 中列出了$format 格式化字符串参数的说明。

表 7-3　$format 参数说明

值	说明	返回值
D	月份中的第几天，有前导零的 2 位数字	01 到 31
D	星期中的第几天，文本表示，3 个字母	Mon 到 Sun
J	月份中的第几天，没有前导零	1 到 31
L	L 的小写字母，表示星期几的完整文本格式	Sunday 到 Saturday
N	数字表示的星期中的第几天	1（表示星期一）到 7（表示星期天）

续表

值	说明	返回值
S	每月天数后面的英文后缀，2 个字符	st，nd，rd 者 th。可以和 j 一起用
W	星期中的第几天，数字表示	0（表示星期天）到 6（表示星期六）
Z	年份中的第几天	0 到 366
W	年份中的第几周，每周从星期一开始	例如：42（当年的第 42 周）
F	月份，完整的文本格式，例如 January 或者 March	January 到 December
M	数字表示的月份，有前导零	01 到 12
M	3 个字母缩写表示的月份	Jan 到 Dec
N	数字表示的月份，没有前导零	1 到 12
t	给定月份所应有的天数	28 到 31
L	是否为闰年	如果是闰年为 1，否则为 0
o	年份数字	例如：2011
Y	4 位数字完整表示的年份	例如：2011
y	2 位数字表示的年份	例如：11
a	小写的上午和下午值	am 或 pm
A	大写的上午和下午值	AM 或 PM
g	小时，12 小时格式，没有前导零	1 到 12
G	小时，24 小时格式，没有前导零	0 到 23
h	小时，12 小时格式，有前导零	01 到 12
H	小时，24 小时格式，有前导零	00 到 23
i	有前导零的分钟数	00 到 59
s	秒数，有前导零	00 到 59
e	时区标识	例如：UTC，GMT，Atlantic/Azores
I	是否为夏令时	如果是夏令时为 1，否则为 0
O	与格林威治时间相差的小时数	例如：+0200
Z	时差偏移量的秒数	-43200 到 43200
c	ISO 8601 格式的日期	2011-05-12T15:19:21+00:00
r	RFC 822 格式的日期	例如：Thu, 21 Dec 2011 06:07:08 +0200
U	从 January 1 1970 00:00:00 开始至今的秒数	与 time()函数相同

【实践案例 7-12】

根据表 7-3 中给出的格式化参数值，编写一个案例演示如何使用 date()函数获取常用格式的日期。

创建一个 date.php 文件，编写如下代码：

```
<?php
$today = date("m/d/y");
echo 'date("m/d/y") 结果为：'.$today;
$today = date("F j, Y, g:i a");
echo '<br/>date("F j, Y, g:i a") 结果为：'.$today;
$today = date("Y 年 n 月 j 日");
echo '<br/>date("Y 年 n 月 j 日") 结果为：'.$today;
$today = date("Y-m-d H:i:s");
echo '<br/>date("Y-m-d H:i:s") 结果为：'.$today;
$today = date("D M j G:i:s T Y");
```

```
echo '<br/>date("D M j G:i:s T Y") 结果为: '.$today;
$today = date('i t  i s  t h e jS  d a y.');
echo "<br/>date('i t  i s  t h e jS  d a y') 结果为: ".$today;
$today = date('\i\t \i\s \t\h\e jS \d\a\y');
echo "<br/>date('\i\t \i\s \t\h\e jS \d\a\y') 结果为: ".$today;
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');
echo "<br/>date('H:m:s \m \i\s\ \m\o\n\t\h') 结果为: ".$today;
?>
```

执行后的输出结果如下所示：

```
date("m/d/y") 结果为: 07/19/12
date("F j, Y, g:i a") 结果为: July 19, 2012, 9:07 am
date("Y年n月j日") 结果为: 2012年7月19日
date("Y-m-d H:i:s") 结果为: 2012-07-19 09:07:32
date("D M j G:i:s T Y") 结果为: Thu Jul 19 9:07:32 CST 2012
date('i t i s t h e jS d a y') 结果为: 07 31 07 32 31 09 Asia/Shanghai 19th
19 am 12.
date('\i \i\s \he jS \d\a\y') 结果为: it is the 19th day
date('H:m:s \m \i\s\ \m\o \h') 结果为: 09:07:32 m is month
```

2. gmdate()函数

gmdate()函数实现与 date()函数相同的功能，不同的是 gmdate()函数返回的时间是格林威治标准时间（GMT）。该函数语法格式如下：

```
string gmdate ( string $format [, int $timestamp ] )
```

例如，下面的代码演示了 gmdate()函数的使用。

```
<?php
echo '今天的星期是: '.gmdate("l");
echo '<br/>今天的日期和时间: '.gmdate('l dS \of F Y h:i:s A');
echo "<br/>2012年12月25日的星期是: " . gmdate("l", mktime(0, 0, 0, 12, 25,
2012));
echo '<br/>gmdate(DATE_ATOM, mktime(0, 0, 0, 5, 1, 2012)) 结果为: '
.gmdate(DATE_ATOM, mktime(0, 0, 0, 5, 1, 2012));
?>
```

执行后的输出结果如下所示：

```
今天的星期是: Thursday
今天的日期和时间: Thursday 19th of July 2012 01:20:16 AM
2012年12月25日的星期是: Monday
gmdate(DATE_ATOM, mktime(0, 0, 0, 5, 1, 2012)) 结果为: 2012-04-
30T16:00:00+00:00
```

3. getdate()函数

getdate()函数用于获取指定的日期和时间信息，语法格式如下：

```
array getdate ([ int $timestamp ] )
```

如果没有指定时间戳$timestamp，则使用系统当前的本地时间。getdate()函数返回一个数组，数组中的每个元素代表日期和时间中的一个特定组成部分。表 7-4 列出了返回数组中键名的说明。

表 7-4　getdate()函数返回数组键名说明

键名	说明	返回值例子
seconds	秒的数字表示	0 到 59
minutes	分钟的数字表示	0 到 59
hours	小时的数字表示	0 到 23
mday	月份中第几天的数字表示	1 到 31
wday	星期中第几天的数字表示	0（表示星期天）到 6（表示星期六）
mon	月份的数字表示	1 到 12
year	4 位数字表示的完整年份	例如：2012
yday	一年中第几天的数字表示	0 到 365
weekday	星期几的完整文本表示	Sunday 到 Saturday
month	月份的完整文本表示	January 到 December
0	自从 UNIX 纪元开始至今的秒数	和 time()的返回值类似是一个时间戳

根据表 7-4 中给出的说明，编写程序输出 getdate()函数的返回值，查看键和值的对应关系。编写实现代码最终如下所示：

```
<?php
$today = getdate();
print_r($today);
printf("今天是%s 年%s 月%s 日  星期%s \n",$today["year"],$today["mon"],
$today["mday"],$today["wday"]);
printf("现在时间是%s 时%s 分%s 秒",$today["hours"],$today
["minutes"],$today["seconds"])
?>
```

执行后的输出结果如下所示：

```
Array
(
    [seconds] => 15
    [minutes] => 37
    [hours] => 9
    [mday] => 19
    [wday] => 4
    [mon] => 7
    [year] => 2012
```

```
    [yday] => 200
    [weekday] => Thursday
    [month] => July
    [0] => 1342661835
)
今天是 2012 年 7 月 19 日 星期 4
现在时间是 9 时 37 分 15 秒
```

4. checkdate()函数

checkdate()函数用于检查一个日期是否有效，有效则返回 true，否则返回 false。该函数的语法格式如下：

```
bool checkdate ( int $month , int $day , int $year )
```

当满足如下的条件时，checkdate()函数将认为是一个有效日期。

- **$year 的值是从 1 到 32767。**
- **$month 的值是从 1 到 12。**
- **$day 的值在给定的$month 所应该具有的天数范围之内，包括闰年的情况。**

例如，下面的代码演示了 checkdate()函数的使用：

```
<?php
echo checkdate(2,29,2012) ? "有效" : "无效";          //返回 true，输出有效
echo checkdate(2,29,2011) ? "有效" : "无效";          //返回 false，输出无效
?>
```

7.3.3 时间函数

PHP 提供的时间函数中最常用的有：time()、mktime()、strtotime()和 microtime()。

1. time()函数

time()函数使用方法非常简单，没有参数调用后返回当前日期和时间的 UNIX 时间戳。Time()函数语法格式如下：

```
int time ( void )
```

下面代码演示了 time()函数的用法：

```
<?php
$nextMonth = time() + (50 * 24 * 60 * 60);      //50 天*24 小时*60 分*60 秒
echo '今天的日期是: '. date('Y-m-d') ."\n";
echo '50 天后的日期是: '. date('Y-m-d', $nextMonth) ."\n";
?>
```

执行后的输出结果如下：

```
今天的日期是：2012-07-19
50 天后的日期是：2012-09-07
```

2. mktime()函数

mktime()函数的作用与 time()函数相同，都可以返回一个时间戳表示的时间。不同的是在 mktime()函数中可以指定具体的日期和时间。mktime()函数语法格式如下：

```
int mktime ([ int $hour ] , [int $minute ] , [int $second] , [int $month ] ,
[int $day ] , [int $year ] )
```

上述参数列表按从右向左顺序省略，任何省略的参数会被设置为本地日期和时间的当前值。如果指定的时间无法表示，则返回 false。

【实践案例 7-13】

mktime()在做日期计算和验证方面很有用，下面的代码用于统计从 2012 年 10 月 1 日到 2012 年 12 月 25 日所经过的天数。

```
<?php
$day1=mktime(0,0,0,12,25,2012);          //表示 2012 年 12 月 25 日
$day2=mktime(0,0,0,10,1,2012);           //表示 2012 年 10 月 1 日
$days=($day1-$day2)/(24 * 60 * 60);      //根据时间戳计算出天数
echo date("y-m-d",$day2)."到".date("y-m-d",$day1)."共差".$days."天";
?>
```

由于 mktime()函数的返回值是时间戳，所以还需要进行转换才能表示天数。执行结果如下：

```
12-10-01 到 12-12-25 共差 85 天
```

3. strtotime()函数

strtotime()函数用于将可阅读的英文日期/时间字符串转换成 UNIX 时间戳。该函数语法格式如下：

```
int strtotime ( string $time [, int $now] )
```

$time 参数表示要被转换的字符串，$now 表示计算返回值的时间戳。如果没有提供$now 参数将使用系统的当前时间。strtotime()函数成功时返回时间戳，否则返回 false。

下面的代码演示了 strtotime()函数的使用：

```
<?php
  echo 'strtotime("now") 结果为: '.strtotime("now")."<br/>";
  echo 'strtotime("10 January 2012") 结果为: '.strtotime("10 January
  2012")."<br/>";
  echo 'strtotime("+1 day" )结果为: '.strtotime("+1 day")."<br/>";
  $str = 'tomorrow after tomorrow';                    //错误的字符串
  //$str = '+1 day';                                   //正确的字符串
```

```
    if ( strtotime($str) == false) {
        echo "运行出错，指定的参数：($str)无法转换为时间。";
        } else {
            $timestamp = strtotime($str);
            echo "$str == " . date('l dS of F Y h:i:s A', $timestamp);
    }
?>
```

执行后的输出结果如下所示：

```
strtotime("now") 结果为：1342666416
strtotime("10 January 2012") 结果为：1326124800
strtotime("+1 day" 结果为：1342752816
运行出错，指定的参数：(tomorrow after tomorrow)无法转换为时间。
```

4．microtime()函数

microtime()调用后将返回当前 UNIX 时间戳和微秒数。该函数语法格式如下：

```
mixed microtime ( [bool $get_as_float] )
```

这里的可选参数$get_as_float 是一个布尔值，如果为 true 将返回一个浮点数。默认不带参数时将以“msec sec”格式返回一个字符串，其中 msec 是微秒部分，sec 是 UNIX 时间戳。

下面的代码演示了 microtime()函数的使用：

```
<?php
echo microtime()."\n";
//输出结果类似 0.26562600 1342667055，0.26562600 表示 msec，1342667055 表示 sec
echo microtime(false)."\n";
//输出结果类似 0.26677000 1342667055 ，0.26677000 表示 msec，1342667055 表示 sec
echo microtime(true)."\n";
//输出结果类似 1342667055.267
?>
```

7.4 XML

XML 是 Internet 环境中跨平台的、依赖于内容的技术，是当前处理结构化文档信息的有力工具。因为 XML 是一种简单的数据存储语言，使用一系列简单的标记描述数据，而这些标记可以用简单的文本格式存储和建立，也可以被任何语言读取和使用。

本节将详细讲解 PHP 如何生成 XML 文档，以及 PHP 提供的 3 种 XML 解析方式。

7.4.1 了解 XML 的结构

XML 全称为 Extensible Markup Language（可扩展标记语言），是一种可扩展标记的语

言，具有自描述性、内容与显示分离、可扩展性、独立于平台等特点。

XML 具有很高的灵活性，但是必须遵循一定的结构，才能保证 XML 文档具有规范性。XML 文档结构可以分为 3 个部分：序言、主体和尾声。其中，序言和主体是每个 XML 文档必须保留的，而尾声则可以根据用户的需要来使用。

1. 序言

序言是 XML 文档的开始，用来表示对 XML 数据进行编译的开始及描述字符的编码方法，并为 XML 解析器和应用程序提供其他一些配置线索。序言部分格式代码如下：

```
<?xml version="1.0" encoding="编码" standalone="yes/no"?>
```

格式中各参数含义如下。

- **version　指定所采用的 XML 版本号，通常情况下，目前 Web 支持的是 XML 1.0 的版本。**
- **encoding　指定文档的编码格式。常用的编码格式有 UTF-8 和 GB2312 两种。在 XML 文档中未指定编码格式的情况下，默认为 UTF-8 格式。使用 UTF-8 格式，标记以及标记的内容就可以使用中文、日文、英文等，XML 解析器就会识别这些标记并正确解析标记中的内容，使用 GB2312 格式，则标记以及标记的内容只可以使用 ASCII 字符和中文。**
- **standalone　表示使用的 DTD 的形式，值只能是 yes 或 no，默认值为 no。如果为 yes，说明所有必需的实体声明都包含在文档中；如果是 no，则说明需要引用外部实体。**

2. 主体

主体就是 XML 文档描述数据的地方，通常由标记、元素和属性等构件组成。

3. 尾声

XML 文档的尾声部分内容包括注释、处理指令和/或紧跟元素树后的空白。由于大多数应用程序在文档根元素的结束标记处就结束了，而不再对尾声进行任何处理，所以尾声部分对于 XML 文档来说不起任何作用。用户可以根据需要选择是否添加尾声。

例如，下面是一个符合 XML 规范的文档：

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
    <Book id="HN330">
        <Title>十万个为什么</Title>
        <Publisher>清华大学出版社</Publisher>
    </Book>
    <Book id="HN331">
        <Title>一千零一夜</Title>
        <Publisher>人民邮电出版社</Publisher>
```

```
    </Book>
</Books>
```

在这个 XML 中第一行是它的序言，剩下的是主体，不包括尾声。在<Books>开始标记和</Books>结束标记之间的为数据，从<Books>开始标记到</Books>结束标记又可以称为 Books 元素。在 XML 主体中<Books>是根元素，其中包含两个 Book 子元素。每个 Book 元素的结构都相同，包含一个 Title 元素和一个 Publisher 元素。

在 XML 主体中除了 XML 元素之外，还包含 XML 属性。XML 属性是指在标记内使用“属性名称=值”形式的代码，本例中的 id 就是 Book 标记的属性。

XML 文档的扩展名为.xml，可以在任何文本编辑器中修改。将上述代码保存为 books.xml，在浏览器中查看效果如图 7-2 所示。

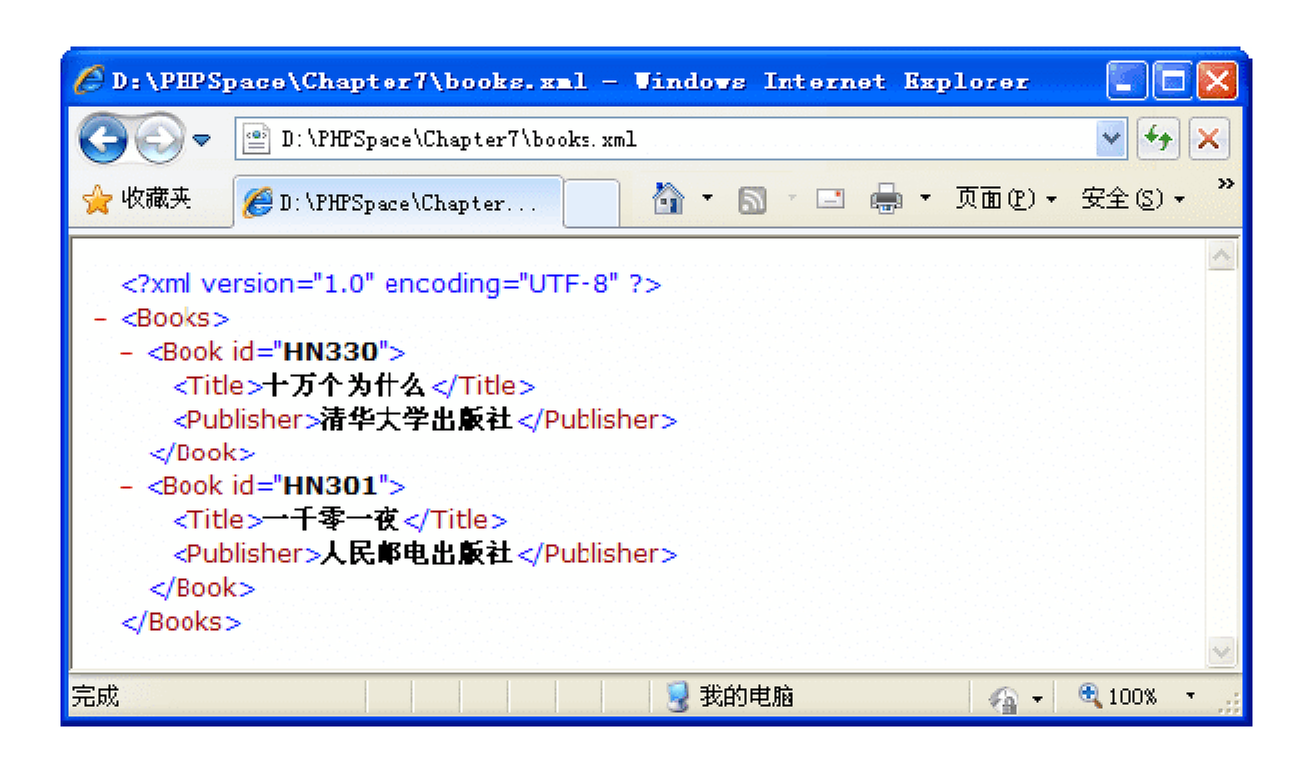


图 7-2　book.xml 运行效果

7.4.2　创建一个 XML 文档

在 7.4.1 节中对 XML 文档的基本结构进行了介绍，并创建了一个 XML 文档。本节介绍如何在 PHP 中创建一个 XML 文档。

PHP 可以动态生成 HTML，当然也可以生成 XML。在生成 XML 时，数据通常都是读取数据库中的。在这里为了演示的方便使用数组来代替，其实现方式都一样。下面通过一个示例演示如何从数组生成 XML。

【实践案例 7-14】

假设有一个数组保存了用户的收藏夹信息，包含网站名称、中文名称和网站地址。现在要从数组中读取数据并以 XML 格式显示到页面上。

（1）创建一个名为 xml.php 的实例文件，并声明两个变量定义收藏夹名称和描述。

```
<?php
$user="均焘";                     //定义收藏夹名称
$desc="我的个人收藏夹";            //定义收藏夹描述
?>
```

（2）接下来创建一个数组保存收藏夹中的网站名称、中文名称和网站地址，如下所示是本示例中的代码：

```
<?php
$sites=array(                          //在数组中定义 XML 的内容
        array(
            'name'=>"baidu",
            'cn name'=>"百度",
            'url'=>"www.baidu.com"),
        array(
            'name'=>"google",
            'cn_name'=>"谷歌",
            'url'=>"www.google.com"),
        array(
                'name'=>"itzccn",
                'cn name'=>"窗内网",
                'url'=>"www.itzcn.com"),
        );
?>
```

（3）在 PHP 中声明为 XML 格式，然后指定编码集，具体实现代码如下所示：

```
<?php
header("Content-Type: text/xml");               //声明为 XML 格式
echo "<?xml version='1.0' encoding='utf-8'?>"; //指定 XML 的声明和编码
echo "<sites>";
echo "<name>{$user}</name>\r\n";
echo "<title>{$desc}</title>\r\n";
foreach($sites as $site) {                      //遍历数组输出 XML 格式内容
    echo "<site>\r\n";
    echo "<name>{$site['name']}</name>\r\n";
    echo "<cn_name>{$site['cn_name']}</cn_name>\r\n";
    echo "<url>{$site['url']}</url>\r\n";
    echo "</site>\r\n";
}
echo "</sites>";
?>
```

如上述代码所示，PHP 动态生成 XML 文档并不复杂。只需要用 header()函数将文档的 MIME 类型改成“"text/xml"”，以此来声明输出的是 XML 格式。然后指定 XML 的根节点 sites，再对数组进行遍历输出 site 节点，每个 site 节点又包含 name、cn_name 和 url 节点。

（4）最后在浏览器中执行 xml.php 文件，会生成如图 7-3 所示的 XML 文档。

7.4.3 SAX 解析 XML

SAX（Simple API for XML）是一个基于事件驱动型的 XML 解析模型，适用于对 XML 文档的遍历，不需要修改内容的情况。使用 SAX 时，每次打开或关闭一个标记，或者每次解析器到文本时，就执行节点或文本的事件处理函数。图 7-4 中给出了使用 SAX 解析 XML

时的流程。

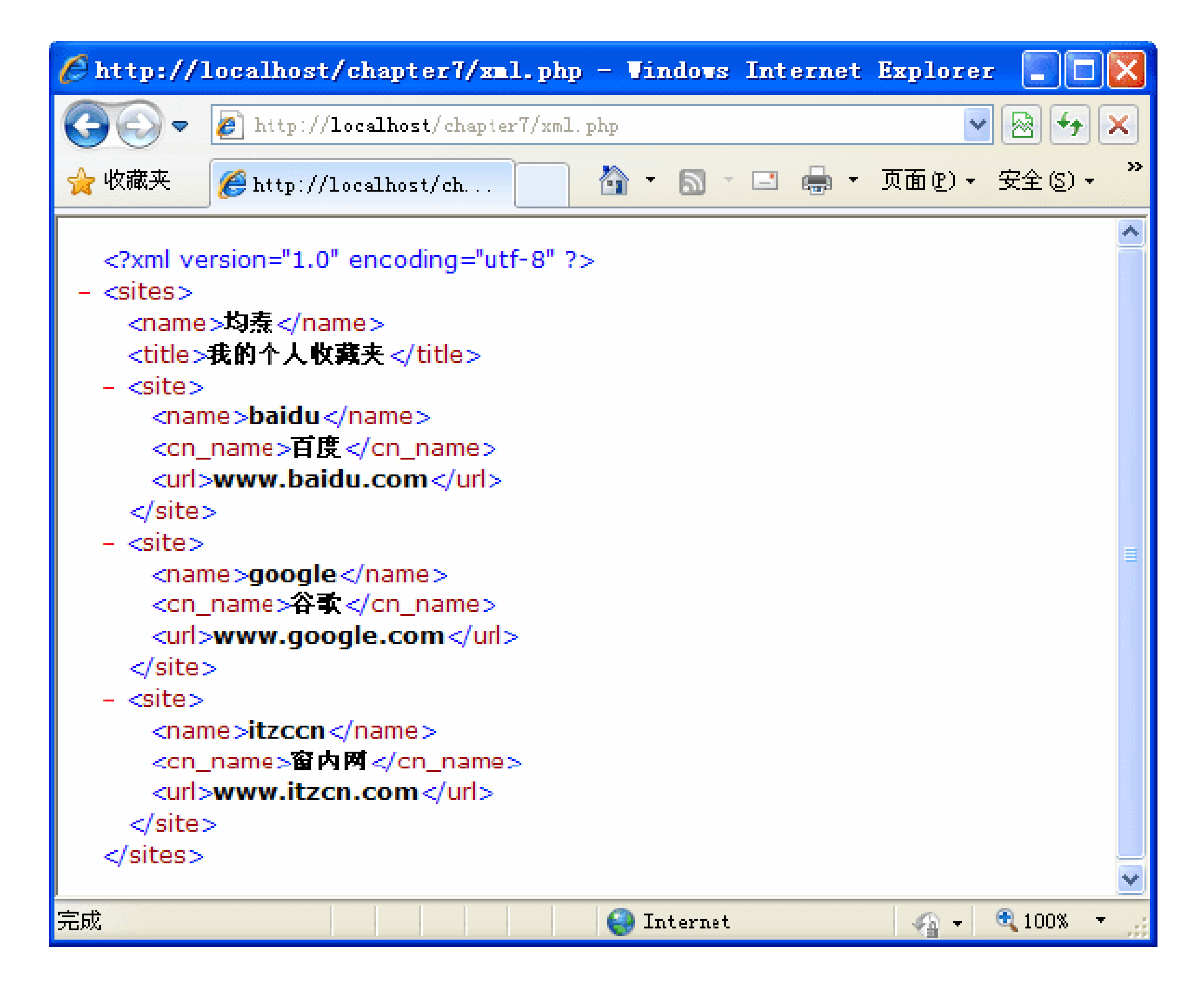


图 7-3　从数组中生成 XML 文档

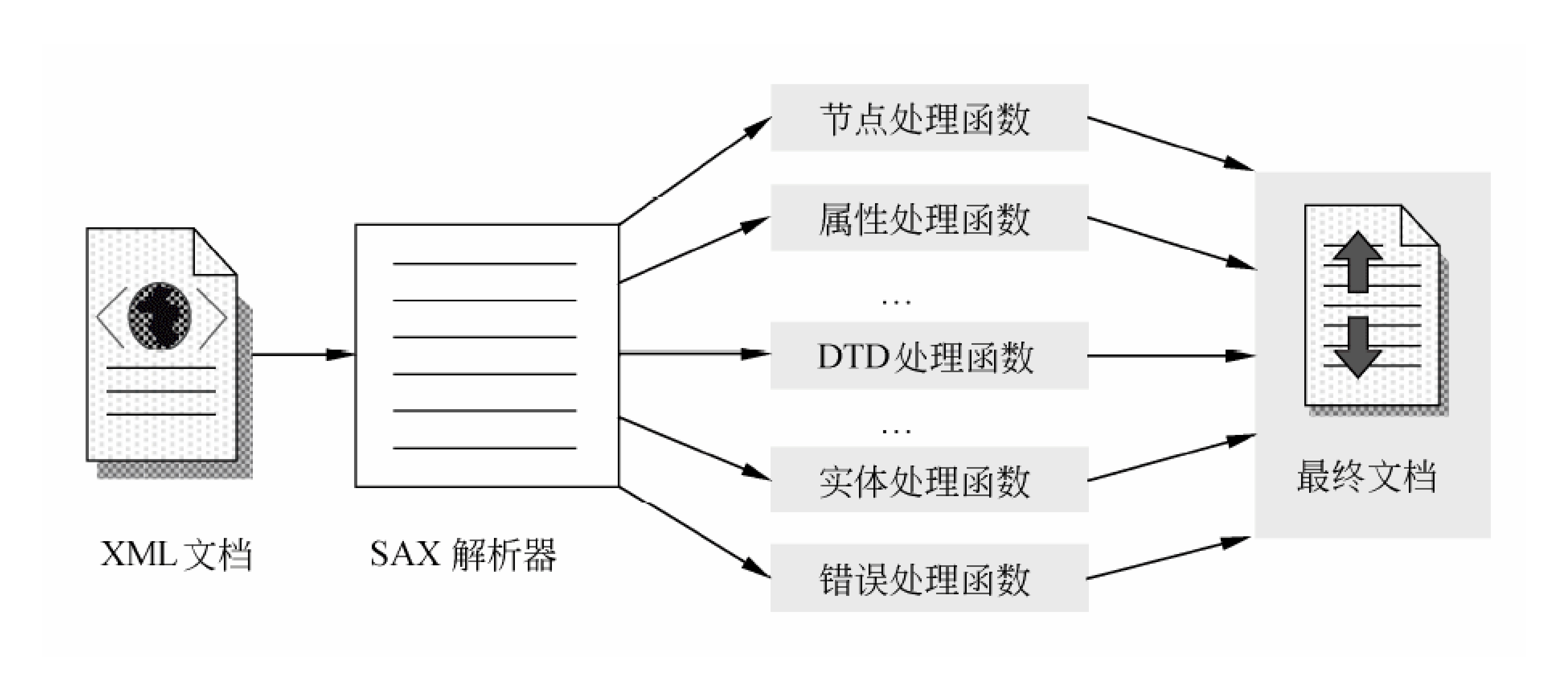


图 7-4　SAX 解析 XML 流程

当使用 SAX 解析器 XML 文档时它将针对各种事件调用不同的处理程序函数。下面介绍解析过程中常用的解析器。

1. 创建 SAX 解析器

要使用 SAX 解析 XML 文档，首先需要创建一个解析器。在 PHP 中使用 xml_parser_create()函数创建 XML 解析器。语法格式如下所示：

```
xml_parser_create(string $encoding)
```

xml_parser_create()函数将建立一个新的 XML 解析器，并返回可被其他 XML 函数使用的资源句柄。$encoding 参数为可选参数，该参数表示指定解析后输出数据的编码，可以是 UTF-8 或者 GBK 等编码。

创建 SAX 解析器之后便可以使用 xml_parse()函数解析 XML 文档，此时将触发各种事件处理器。数据处理完毕后调用 xml_parser_free()函数释放解析器。xml_parse()函数语法格

式如下所示：

```
int xml_parse ( resource $parser , string $data [, bool $is_final ] )
```

其中，$data 参数是要处理的 XML 字符串，为了使最后一块数据被解析，可选的$final 参数应设为 true。如果解析器成功解析 XML 将返回 true，否则返回 false。



可以使用 xml_get_error_code() 函数获取错误的编号，再使用 xml_error_string()函数将错误编号转换成字符串。

2. 元素处理器

xml_set_element_handler()函数可以设置元素处理器，语法格式如下所示：

```
bool xml set element handler ( resource $parser , callback $start
element_handler , callback $end_element_handler )
```

其中，$parser 参数表示要使用的 XML 解析器，$start_element_handler 参数和 $end_element_handler 参数表示处理器函数的名称，并且这两个函数都是按引用传递的。当 XML 解析器遇到元素的开始标签时，会调用 start_element_handler 元素处理函数，该函数必须有 3 个参数，其语法格式如下所示：

```
start_element_handler ( resource $parser , string $name , array $attribs )
```

这里的 3 个参数分别表示：调用处理程序的 XML 解析器引用、开始元素的名称和解析器遇到元素的属性数组。

当 XML 解析器遇到元素的结束标签时，会调用 end_element_handler 元素处理函数，该函数必须有两个参数，其语法格式如下所示：

```
end_element_handler ( resource $parser , string $name )
```

这里的$parser 参数表示调用处理程序的 XML 解析器引用，$name 参数为被调用的元素名。

3. 字符数据处理器

元素之间所有文本由字符数据处理器处理。在 PHP 中，通过 xml_set_character_data_handler()函数设置遇到每一个字符数据块时调用的处理器，语法格式如下所示：

```
bool xml_set_character_data_handler ( resource $parser , callback $handler )
```

其中，$parser 参数表示触发处理程序的 XML 解析器的引用，$handler 参数表示作为事件处理器使用的函数。

由$handler 参数指定的函数必须有两个参数，语法格式如下：

```
handler ( resource $parser , string $data )
```

例如，下面是一个$handler 事件处理器函数的示例代码：

```
function char($parser,$data)
{
  echo $data;            //输出节点文本
}
```

4. 默认处理器

只要能在 XML 文档中找到数据，就可以调用 xml_set_default_handler()函数解析 XML 文档。xml_set_default_handler()函数为 XML 解析器建立默认的数据处理器，如果成功创建处理器返回 true，否则返回 false。函数语法格式如下所示：

```
bool xml_set_default_handler ( resource $parser , callback $handler )
```

这里的$parser 参数表示要使用的 XML 解析器，$handler 参数表示事件处理器使用的函数。

下面的代码演示了如何使用 xml_set_default_handler()函数解析 XML：

```
<?php
$parser=xml_parser_create();                       //新建解析器
function mydefault($parser,$data)                  //创建默认处理器函数
{
   echo $data;
}
xml_set_default_handler($parser,"mydefault");      //指定默认处理器
$fp=fopen("newslist.xml","r");                     //打开 XML 文档
while ($data=fread($fp,4096))                      //读取 XML 文档
{
  xml parse($parser,$data,feof($fp))               //解析 XML
}
xml_parser_free($parser);                          //释放 XML 解析器
?>
```

7.4.4 DOM 解析 XML

DOM（Document Object Model，文档对象模型）解析 XML 时，需要把整个 XML 文件加载到内存中去，并以一棵节点树的形式存在。当加载完成后，可以对节点树中的每一个节点进行操作。换句话说，通过 DOM 应用程序可以对 XML 文档进行随机访问。这种访问方式给应用程序的开发带来了很大的灵活性，它可以任意地控制整个 XML 文档中的内容，从而执行添加、删除、修改等操作。

例如，下面所示为一个简单的 XML 文档：

```
<?xml version="1.0" encoding="gb2312" ?>
```

```
<Books>
    <Book>
        <id>0515</id>
        <title>唐诗三百首</title>
        <price>6.8</price>
    </Book>
</Books>
```

上述所示就是一个由一个节点组成的 DOM 树结构。其中，Books 是文档的根节点，在根元素里面包含一个 Book 子节点；而 Book 又包含 3 个子节点：id、title 和 price，并且都包含了相应的数据信息。在 DOM 中，每一个子节点都会成为一个节点，属性也是一个节点，标记间的数据也是一个节点，可以清晰地表达出文档中各节点之间的关系，图 7-5 所示为上述 XML 的文档树。

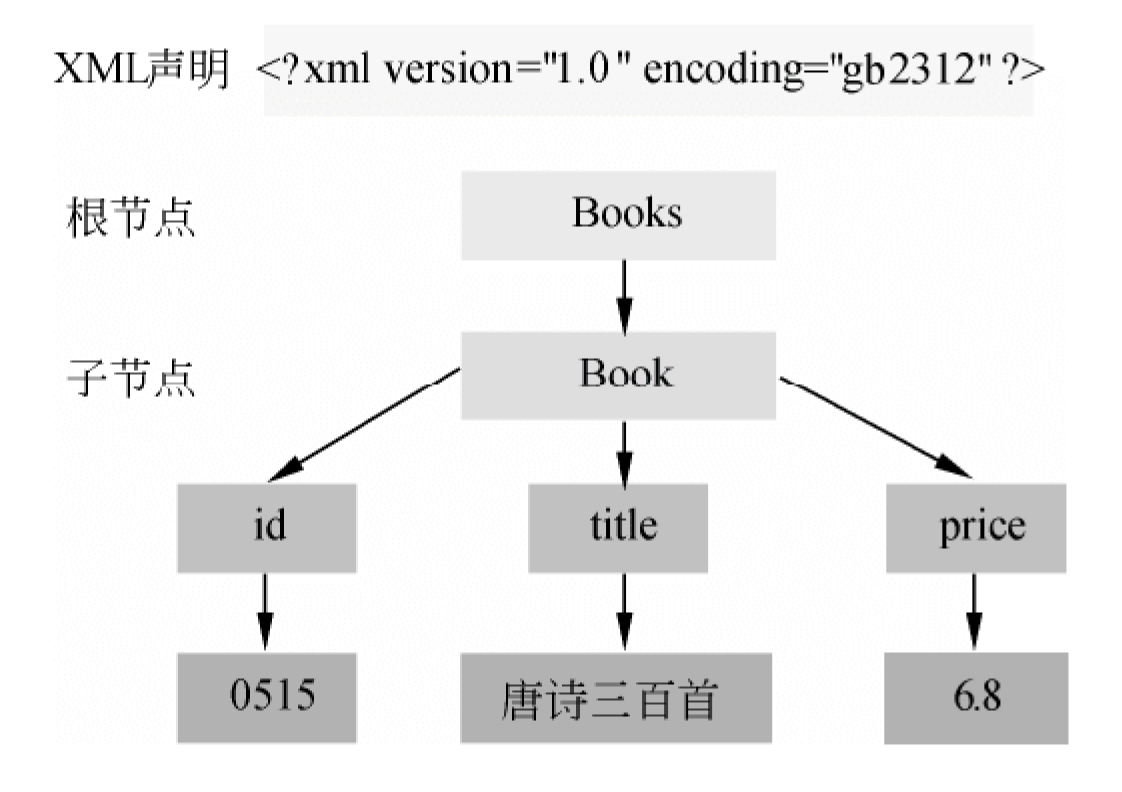


图 7-5　DOM 解析 XML 流程

DOM 提供了很多用于解析时从 XML 中获取数据的函数，在表 7-5 中列出了最常用的函数及其说明。

表 7-5　常用 XML 操作函数

函数名称	说明
DOMDocument()	创建一个 DOM 对象
load('name.xml')	加载 XML 文件 name.xml 到内存
getElementsByTagName("book")	获取当前节点下 book 节点的列表对象
item(index)	获取 index 指定索引值的节点对象
nodeValue()	获取节点的值
getAttribute("name")	获取当前节点 name 属性的值

【实践案例 7-15】

下面以 7.4.1 节创建的图书 XML 文档为例，讲解如何通过 DOM 方式进行解析。

（1）在图书 XML 文档 books.xml 的目录中创建一个实例文件，名称为 dom_xml.php。

（2）利用表 7-5 中提供的函数，先从 XML 文档中读取网站的名称和地址信息。

```
<h1>儿童读物</h1>
<table  cellpadding="0"  cellspacing="0">
```

```
<tr>
    <th class="first">编号</th>
    <th>图书名称</th>
    <th>价格</th>
</tr>
<?php
$doc = new DOMDocument();                           //创建 DOM 对象
$doc->load("books.xml");                            //载入外部的 XML 文件
$books= $doc->getElementsByTagName("Book" );        //获取 Book 节点列表
foreach( $books as $book )                          //循环读取每个节点的信息
{
    $idText =$book->getAttribute('id');             //获取 id 属性的值
    $booktitle = $book->getElementsByTagName("Title");
    $titleText =$booktitle->item(0)->nodeValue;//获取 title 节点的值
    $bookpub = $book->getElementsByTagName( "Publisher" );
    $pubText =$bookpub->item(0)->nodeValue;         //获取 Publisher 节点的值
    ?>
<tr class="bg">
    <td class="first style1"><?php echo $idText; ?></td>
    <td><?php echo $titleText; ?></td>
    <td><?php echo $pubText; ?></td>
</tr>
<?php } ?>
</table>
```

在上述代码中，首先用“$doc = new DOMDocument()”创建了一个 DOM 对象的实例 $doc，然后调用 load()函数载入 books.xml。语句“$doc->getElementsByTagName("Book")”从 XML 文档中搜索 Book 节点并返回，再往下的“$book->getAttribute('id')”语句则获取了 Book 节点 id 属性的值。

语句“$doc->getElementsByTagName("Book")”返回的是一个包含所有 Book 节点的集合。所以，下面使用 foreach()语句循环该集合，再依次获取每个节点的值，最后在表格中输出。

（3）将 PHP 文件和 XML 文件放在同一目录。然后浏览 dom_xml.php 文件查看读取 XML 文档的效果，如图 7-6 所示。

除了上例中遍历节点输出 XML 文档的内容外，DOM 还提供了一种快速输出内容的方法，那就是调用 saveXML()函数。示例代码如下所示：

```
<?php
$doc = new DOMDocument();                //创建 DOM 对象
$doc->load( 'newslist.xml' );            //载入 XML 文档
print $doc->saveXML();                   //输出 XML 文档的内容
?>
```

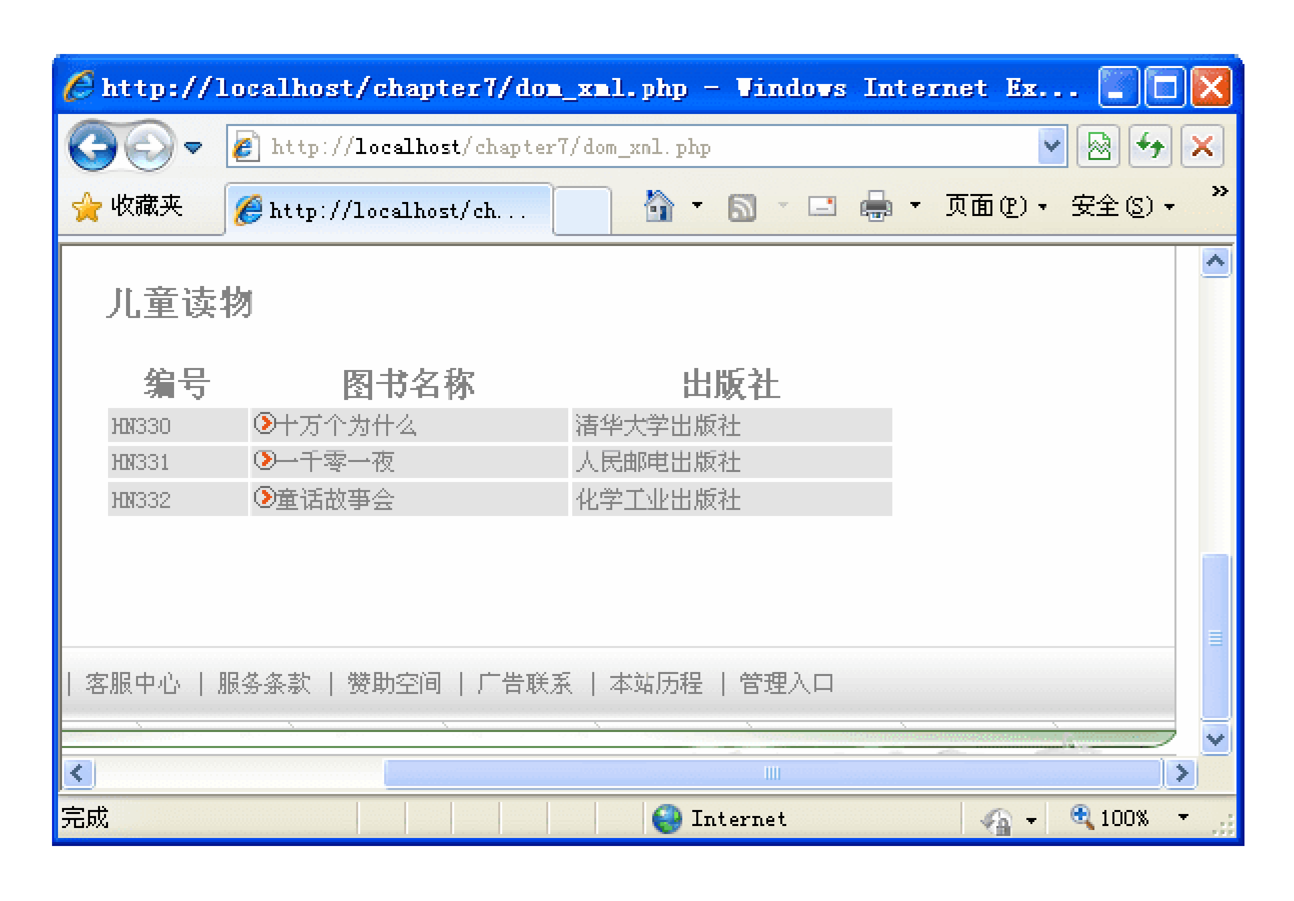


图 7-6　读取 XML 文档运行效果

【实践案例 7-16】

DOM 解析器的功能非常强大，除了上面介绍的遍历 XML 文档的方法之外，使用 DOM 创建一个 XML 文档同样比较简单。下面代码就创建了 7.4.1 节的图书 XML 文档。

```
<?php
$dom=new DOMDocument("1.0", "utf-8");              //创建 DOM 解析器对象

$root=$dom->createElement("Books");                //创建根节点 Book
$book=$dom->createElement("Book");                 //创建 Book 节点
$title=$dom->createElement("Title");               //创建 Title 节点
$publisher=$dom->createElement("Publisher");       //创建 Publisher 节点
$id=$dom->createAttribute("id");                   //创建 id 属性

$idText=$dom->createTextNode("HN330");             //创建 id 属性的值
$titleText=$dom->createTextNode("十万个为什么");    //创建 Title 节点的值
$pubText=$dom->createTextNode("清华大学出版社");    //创建 Publisher 节点的值

$id->appendChild($idText);                 //将值添加到 id 节点
$publisher->appendChild($pubText);         //将值添加到 Publisher 节点
$title->appendChild($titleText);           //将值添加到 Title 节点

$book->appendChild($id);                   //向 Book 节点中添加 id 属性
$book->appendChild($title);                //向 Book 节点中添加 Title 节点
$book->appendChild($publisher);            //向 Book 节点中添加 Publisher 节点

$root=$dom->appendChild($root);            //生成根节点 Books
$book=$root->appendChild($book);           //生成 Books 下的 Book 节点
$dom->save("Books1.xml");                  //保存到 Books1.xml 文件
?>
```

上述代码执行后将创建一个 Books1.xml 文件，其中保存了生成的 XML 内容。

7.4.5 SimpleXML 解析 XML

SimpleXML 是 PHP 5 核心的一部分，因此无需安装或引用外部扩展，就可以对 XML 文档进行处理。

SimpleXML 也是解析和操作简单 XML 文档的最佳解析器。它会把 XML 转换为对象，然后通过属性提供每个节点元素的访问方法，还可以对从任何节点中获取的文本值进行字符串转换。

SimpleXML 会将 XML 文档转换为 SimpleXML 对象，文档中的元素被转换为 SimpleXMLElement 对象的单一属性。当同一级别上存在多个元素时，它们会被置于数组中，通过使用关联数组进行访问属性，下标对应属性名称。如果一个元素拥有多个文本节点，则按照它们被找到的顺序进行排列。

表 7-6 中列出了使用 SimpleXML 解析 XML 文档时最常用的函数及其说明。

表 7-6　SimpleXML 常用函数

函数名称	说明
__construct()	创建一个新的 SimpleXMLElement 对象。如果执行成功，则返回一个对象，否则返回 false
addAttribute()	为 SimpleXML 元素添加一个属性，该函数无返回值
addChild()	给 SimpleXML 元素添加一个子元素
asXML()	从 SimpleXML 元素获取 XML 字符串
attributes()	获取 SimpleXML 元素的属性
children()	获取指定节点的子节点
getDocNamespaces()	获取 XML 文档的命名空间
getName()	获取 SimpleXML 元素的名称
getNamespaces()	从 XML 数据获取命名空间
registerXPathNamespace()	为下一次 XPath 查询创建命名空间语句
simplexml_import_dom()	从 DOM 节点获取 SimpleXMLElement 对象
simplexml_load_file()	从 XML 文档获取 SimpleXMLElement 对象
simplexml_load_string()	从 XML 字符串获取 SimpleXMLElement 对象
Xpath()	对 XML 数据运行 XPath 查询

【实践案例 7-17】

根据表 7-6 中给出的函数使用 SimpleXML 方式解析一个 XML 文档。假设，该 XML 文档名为 NewsList.xml，包含的内容如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<newslist>
    <news isnew="1">
        <id>5</id>
        <title>PHP 开发全套视频教程</title>
        <desc>从零开始学 PHP 的最佳实践，适合初学者</desc>
        <date>2012-08-8</date>
    </news>
```

```
        <news isnew="1">
            <id>6</id>
            <title>本站评论模块上线</title>
            <desc>欢迎大家发表评论公测</desc>
            <date>2012-07-29</date>
        </news>
        <news isnew="0">
            <id>7</id>
            <title>创建 PHP 开发环境</title>
            <desc>基于 LAMP 的搭建方法</desc>
            <date>2012-7-12</date>
        </news>
    </newslist>
```

（1）创建名为 simple_xml.php 的实例文件，与 NewsList.xml 保存到相同目录。

（2）编写代码使用 SimpleXML 载入 XML 文档，然后对其中的节点进行遍历输出 isnew 属性、各个子节点的名称和值，最终代码如下所示：

```
<fieldset><legend>使用 SimpleXML 解析 XML</legend>
 <?php
 $xml = simplexml_load_file("roster.xml");      //载入 XML 文档
 foreach($xml->children() as $art)              //遍历 XML 文档节点
 {
    if($art->getName()=="student") {            //输出 student 节点的 ID 属性
        echo("student 的 ID 属性: ".$art->attributes()->ID."<br/>");
    }
    foreach($art->children() as $item)          //输出 student 节点下的内容
    {
        echo($item->getName()."节点: ".$item."<br/>");   //输出节点名称和值
    }
 }
 echo("<br/>运行 XPath 查询所有的 name 节点: <br/>");
 $byline=$xml->xpath("/roster/student/name");   //运行 XPath 查询
 foreach($byline as $item)
 {
    echo "$item , ";                            //输出节点值
 }
 ?>
 </fieldset>
  <h3>站内公告</h3>
    <table width="500"  cellpadding="2" cellspacing="2">
      <tr>
        <th class="first">编号</th>
        <th>标题</th>
        <th>描述</th>
```

```
        <th>发表日期</th>
      </tr>
      <?php
$xml = simplexml_load_file("NewsList.xml");         //载入 XML 文档
foreach($xml->children() as $news)                  //遍历 XML 文档节点
{
    $hot="";
    if($news->getName()=="news") {                  //获取 isnew 属性
        $hot=$news->attributes()->isnew=="1"?"<img src='images/hot.
        gif'/>":"";
    }
    ?>
   <tr class="bg">
     <td height="32" class="first style1"><?php echo $news->id; ?></td>
     <td><img border="0" src="images/jt1.gif" width="11" height="11"><?php
    echo $news->title; ?><?php echo $hot; ?></td>
     <td><?php echo $news->desc; ?></td>
     <td><?php echo $news->date; ?></td>
   </tr>
   <?php }  ?>
 </table>
```

如上述代码所示，在这里使用到了 simplexml_load_file()函数、children()函数和 attributes()函数等。表 7-6 中对这些方法已经详细讲解，在此就不再解释。

（3）从 simple_xml.php 文件中运行，在页面中查看效果，如图 7-7 所示。

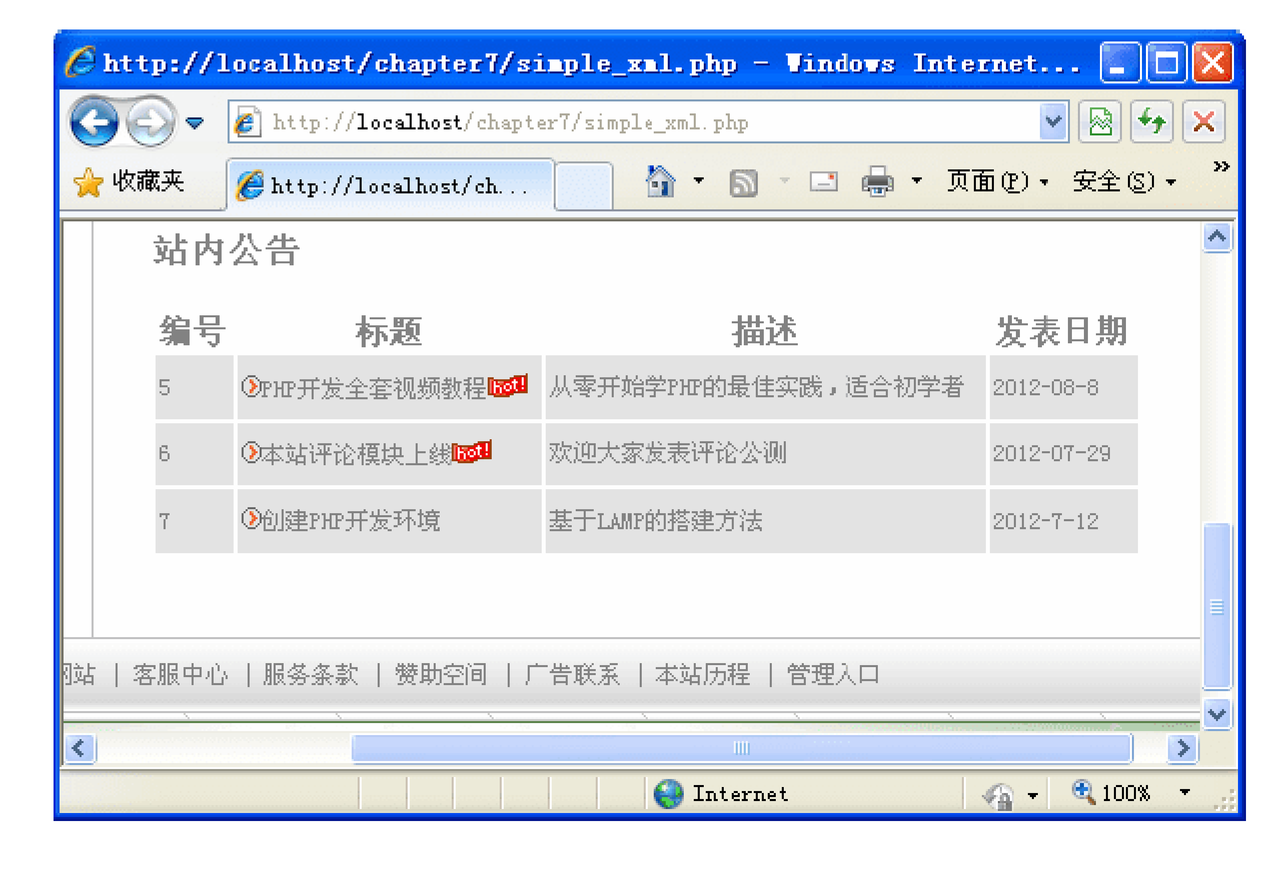


图 7-7　页面效果图

【实践案例 7-18】

与 DOM 的复杂性相比，SimpleXML 就比较简单。不仅可以快速解析 XML 文档，还可以使用对象的方式生成 XML 文档。如下面的示例代码：

```
<?php
//创建一个 SimpleXML 对象的 XML，根节点是 config
```

```
$xml = new SimpleXMLElement('<?xml version="1.0" encoding="utf-8"?>
<config/>');
$xml->addChild("desc","数据库配置文件");   //添加 desc 节点
$db= $xml->addChild('database');          //添加 database 节点
$db->addAttribute('type', 'mysql');       //为 database 节点添加 type 属性
$db->addChild('host', 'localhost');       //为 database 节点添加 host 子节点
$db->addChild('user', 'root');            //为 database 节点添加 user 子节点
$db->addChild('pass', '123456');          //为 database 节点添加 pass 子节点
$xml->asXML("config.xml");                //保存到 config.xml
?>
```

上述代码执行后，将在程序目录创建一个 config.xml 文件。打开之后将看到生成的如下内容：

```
<?xml version="1.0" encoding="utf-8" ?>
<config>
  <desc>数据库配置文件</desc>
  <database type="mysql">
    <host>localhost</host>
    <user>root</user>
    <pass>123456</pass>
  </database>
</config>
```

7.5 正则表达式

正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对字符串的一种过滤逻辑。例如，在网页上填表时经常用到的 Email、电话、密码和生日之类的数据都有特定格式，这个时候就可以使用正则表达式验证数据是否有效。

PHP 支持两种类型的正则表达式：POSIX 类型和 Perl 类型。下面分别介绍每种类型的正则表达式语法及其处理函数。

7.5.1 POSIX 正则表达式语法

POSIX 正则表达式的结构和一般的数学表达式相似，由多个操作符（元素）组合在一起构成一个更复杂的表达式。这种组合不仅可以找到或者匹配表达式(如某个单词或数字)，还可以找到许多语义不同但语法相似的字符串（如 HTML 标记）。

在 POSIX 中支持三种语法来定义正则表达式，分别是中括号、量词和预定义字符。

1. 中括号（[]）

POSIX 正则表达式的中括号（[]）表示在一定的范围内查找字符，可以在中括号内设

置要查找的字符序列。例如，[abc]表示匹配在这 3 个字符之间的字符串。

为了表示方便，可以在中括号内使用短横线连接两个字符来表示一个连续范围，例如[0-9]就表示 0 到 9 之间的数字，还可以用[2-8]和[c-y]等。表 7-7 列出了正则表达式常用的字符范围。

表 7-7 常用的字符范围

常用形式	说明
[0-9]	匹配任何从 0 到 9 之间的十进制数字
[a-z]	匹配任何从小写 a 到 z 之间的字符
[A-Z]	匹配任何从大写 A 到 Z 之间的字符
[A-Za-z]	匹配任何从大写 A 到小写 z 之间的字符

2. 特殊字符

在中括号中指定字符范围时有一个很明显的局限性。例如要判断某个字符串是否为 18 位的身份证号码，是不是就需要“[0-90-90-90-9——0-9]”这样将 0-9 重复 18 次呢？这样写当然可以，但很明显有它的不合理性，如果不是 18 位，而是一千位呢？

所以，POSIX 正则表达式中定义了一些特殊字符和表达式，用于简化类似复杂情况下的正则表达式写法，如表 7-8 所示。

表 7-8 特殊字符

常用形式	说明
p+	匹配任何一个至少包含 p 的字符串
p*	匹配任何包含零个或多个 p 的字符串
p?	匹配任何包含零个或一个 p 的字符串
p{2}	匹配任何包含两个 p 序列的字符串
p{2,3}	匹配任何包含两个或 3 个 p 序列的字符串
p{2.}	匹配任何至少包含两个 p 序列的字符串
p$	匹配任何以 p 结尾的字符串
^p	匹配任何以 p 开头的字符串
[^a-zA-Z]	匹配任何不包含从 a 到 z 和从 A 到 Z 的字符串
p.p	匹配任何包含字符 p、接下来是任何字符串、再接下来又是 p 的字符串
^.{2}$	匹配任何只包含两个字符的字符串
(.*)	匹配任何被和包围的字符串
p(hp)*	匹配任何包含一个 p，后面是零个或多个 hp 的字符串
(a|b)	匹配任何包含 a 或者 b 的字符串

在这里要注意^符号在中括号之外表示字符串的开头，例如^h 可以匹配 hello，但是不以匹配 the。当^符号在中括号之内时表示取反，“非”或者“排除”的意思，常常用于否定某些字符。例如^[^0-9][0-9]表示第 1 个字符不能是数字，第 2 个位是数字的情况。因此，可以匹配-2、H2 或者@2 之类的字符，但不能匹配 02 或者 22 之类的字符。

下面列出了^符号用于排除字符的常规用法。

- [^0-9]：**匹配除了数字以外的所有字符。**
- [^a-z]：**匹配除了小写字母以外的所有字符。**

- [^A-Z]： 匹配除了大写字母以外的所有字符。
- [^\\\/\^]： 匹配除了"\"、"/"和"^"以外的所有字符。
- [^\"\']： 匹配除了双引号和单引号以外的所有字符。

3. 预定义字符

在 POSIX 正则表达式中还可以使用一些预定义的字符表示范围。预定义字符可以指定整个字符范围，例如字母或整数集。表 7-9 列出了常用的预定义字符。

表 7-9 预定义字符

字符	描述	扩展
[:alnum:]	字母和数字字符	[0-9a-zA-Z]
[:alpha:]	字母字符（字母）	[a-zA-Z]
[:ascii:]	7 位 ASCII	[\x01-\x7F]
[:blank:]	水平空白符（空格、制表符）	[\t]
[:cntrl:]	控制字符	[\x01-\x1F]
[:digit:]	数字	[0-9]
[:lower:]	小写字母	[a-z]
[:print:]	可打印字符（图形类加空格和制表符）	[\t\x20-\xFF]
[:punct:]	任意标点符号，如句号（.）和分号（;）	[-!"#$%&'( )*+,./:;<=>?@[\\\]^_'{|}~]
[:space:]	空白（换行、回车、制表符、空格、垂直制表符）	[\n\r\t \x0B]
[:upper:]	大写字母	[A-Z]
[:xdigit:]	十六进制数字	[0-9a-fA-F]

每一个预定义字符都可以被用于替代一类字符。例如，要查找任一个小写字母或者一个数字，可以使用下面的正则表达式：

```
[[:lower:] [:digit:]]
```

但是，不能把一个预定义字符当作一个范围的终点使用。例如，下面的表达式是错误的：

```
ereg('[0-[:alnum:]]', '234');                        //非法的正则表达式
```

7.5.2 POSIX 正则表达式函数

PHP 提供了 4 个专门用于 POSIX 正则表达式的处理函数，分别是 ereg()、eregi()、ereg_replace()和 eregi_replace()。

1. ereg()函数

ereg()函数用于在字符串中查找匹配正则表达式的子字符串，语法结构如下：

```
int ereg (string $pattern, string $string [, array $regs])
```

其中，$pattern 表示正则表达式，$string 表示要查询的字符串。如果定义了可选参数

$regs，那么当找到与$pattern 模式相匹配的子串时，匹配项将被存入$regs 数组中。$regs[1]包含第一个左圆括号开始的子串，$regs[2]包含第二个子串，依次类推，而$regs[0]则包含整个匹配的字符串。

【实践案例 7-19】

例如，要对一个“年-月-日”格式的日期进行判断，其中年份用 4 位数字表示。首先编写用于对这个日期进行测试的正则表达式：

```
([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})     //4 位数字-1 至 2 位数字-1 至 2 位数字
```

然后使用 ereg()函数对日期正则表达式匹配判断，如果符合则输出返回的数组。实现代码如下：

```
<?php
$date =Date("2012-12-25");                          //指定一个日期
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
                                              //使用正则表达进行匹配
    echo "匹配成功，返回的数组如下：\n";
    print_r($regs);                           //输出匹配的数组
} else {
    echo "非法的日期格式：$date";
}
?>
```

将上述代码保存到 ereg.php 文件，执行后将看到如下输出结果：

```
匹配成功，返回的数组是：
Array
(
    [0] => 2012-12-25
    [1] => 2012
    [2] => 12
    [3] => 25
)
```

ereg()函数在匹配时区分字符串的大小写，如果没有找到匹配或出错则返回 false。例如，下面代码从$str 中匹配字母 b，运行后输出“失败”。

```
<?php
$str="Red Black Yellow";    //定义查找字符串
if(ereg("b",$str)){       //查询字母 b，由于 ereg()函数区分大小写，所以返回 false
    echo "成功";
}
else{
    echo "失败";           //输出“失败”
}
?>
```

2. eregi()函数

eregi()函数的语法与 ereg()函数完全相同，唯一不同的是 eregi()函数在匹配时不区分大小写。

【实践案例 7-20】

下面的代码演示了如何使用 eregi()函数验证字符串和邮箱格式是否正确。

```
<?php
$str = "www.itzcn.com";                    //指定一个字符串
if (eregi('ITZCN', $str)) {                //匹配是否有 ITZCN，这里不区分大小写
    echo $str."这个字符串中含有字符'itzcn'。";
}
?>
```

上述代码执行后将看到如下输出结果：

```
www.itzcn.com 这个字符串中含有字符'itzcn'。
```

3. ereg_replace()函数

ereg_replace()函数用于将字符串中与正则表达式相匹配的子字符串替换成指定字符串，函数语法格式如下：

```
string ereg_replace(string $pattern, string r$eplacement, string $string)
```

其中，$pattern 表示正则表达式，$replacement 表示用于替换的字符串，$string 表示原始字符串。ereg_replace()函数在$string 中扫描与$pattern 匹配的部分，并将其替换为$replacement，最后返回替换后的字符串，如果没有可供替换的匹配项则会返回原字符串。

【实践案例 7-21】

下面的代码演示了如何使用 ereg_replace()函数替换一个字符串、一个单词以及一个 URL。

```
<?php
$string = "Saturday Thursday Friday";
echo "替换字符串的结果: \n";
echo str replace("satur", "Mon", $string);          //替换字符串
echo "\n 替换 URL 为超链接的结果: \n";
$text="本站网址 www.itzcn.com";                      //替换 URL
$text = ereg replace("[^<>[:space:]]+[[:alnum:]/]", "<a href=\"http:
//\\0\">\\0</a>", $text);
echo $text;
?>
```

将上述代码保存到 ereg_replace.php 文件，执行后将看到如下输出结果：

```
替换字符串的结果:
Saturday Thursday Friday
替换 URL 为超链接的结果:
本站网址 <a href="http://www.itzcn.com">www.itzcn.com</a>
```

4. eregi_replace()函数

eregi_replace()函数的语法与 ereg_replace()函数完全相同，唯一不同的是 eregi_replace()函数在替换时不区分大小写。

下面的代码演示了如何使用 eregi_replace()函数替换一个字符串。

```
<?php
$pattern = '[A-Z]{5}';
$repStr= "WWW";
$str= "asdfg:Face:gfdsa";
$result = eregi_replace($pattern, $repStr, $str);
echo "替换前: ".$str;
echo "\n 替换后: ".$result;
?>
```

将上述代码保存到 eregi_replace.php 文件，执行后将看到如下输出结果：

```
替换前: asdfg:Face:gfdsa
替换后: WWW:Face:WWW
```

7.5.3 Perl 正则表达式语法

这种正则表达式语法源自 Perl 语言，Perl 也是对字符串操作功能最强大的语言之一。PHP 引入了这种风格的正则表达式语法，下面首先介绍 Perl 风格正则表达式的定义方法。

1. 定界符

Perl 风格正则表达式要求每个模式都必须用一对定界符括起来，习惯上使用一对斜线（/）作为定界符，例如，/pattern/。不过，任意非数字和字母的字符（除了反斜线（\））都可作为定界符，尤其是在匹配的字符串中本身包含斜线时，一般用其他字符作为定界符。

例如，有下面这样的字符串：

```
D:/music/ting
```

可以使用下面的正则表达式匹配该字符串：

```
/D:\/music\/ting/
```

由于定界符使用的是“/”，而字符串中也包含有“/”，所以在正则表达式中，除了定界符以外，其他“/”前面都需要添加反斜线进行转义。此时就应该使用其他字符作为定界符，例如如下的正则表达式使用字符“#”作为定界符，这样就不需要在“/”前面添加反

斜线进行转义。

```
# D:/music/ting#
```

2. 修饰符

在进行正则表达式匹配时，可能需要不区分大小写或者只查找第一个相匹配的字符串等。这些要求可以通过使用 Perl 正则表达式中的修饰符来实现。表 7-10 中列出了 Perl 正则表达式中常用的修饰符。

表 7-10 常用修饰符

修饰符	含义
i	不区分大小写进行匹配
m	将一个字符串视为多行。默认情况下，^和$字符匹配字符串中的最开始和最末尾。使用 m 修饰符将使^和$匹配字符串中每行的开始
s	将一个字符串视为一行，忽略其中的换行符；它与 m 修饰符正好相反
x	忽略正则表达式中的空白和注释
U	第一次匹配后停止，即查找到第一个字符串后，停止后面的搜索

修饰符一般直接放在正则表达式的后面，例如：

```
/Php/i
```

正则表达式/Php/原本只可以匹配字符串 Php，而在使用了修饰符 i 后，就可以匹配字符串 php、PHP 或者 pHP 等。表 7-10 中的多个修饰符还可以同时使用。例如：

```
/P    h    p/ix
```

3. 预定义字符

预定义字符用于设置正则表达式的匹配范围，它总是以一个反斜线“\”开始，后面跟一个字母，表 7-11 中列出了常用的预定义字符。

表 7-11 常用预定义字符

元字符	含义
\A	只匹配字符串开头
\b	匹配单词边界
\B	匹配除单词边界外的任意字符
\d	匹配数字字符，与[0-9]相同
\D	匹配非数字字符
\s	匹配空白字符
\S	匹配非空白字符
\w	匹配任何只包含下划线和字母数字字符的字符串，与[a-zA-Z0-9_]相同
\W	匹配没有下划线和字母数字字符的字符串

每个预定义字符只能匹配一个字符，如果要匹配多个，同样可以使用 POSIX 正则表达式中的特殊字符。例如，下面的正则表达式可以匹配一个或多个数字字符：

```
/\d+/
```

表 7-11 中的常用元字符都比较好理解，下面对\b 与\B 作简单解释。例如，有如下 5 个正则表达式及其解释：

```
/a/          //可以匹配任意包含字符 a 的字符串
/^a/         //可以匹配任意以字符 a 开头的字符串
/a\b/        //可以匹配任意以字符 a 结尾的字符串
/\ba\b/      //可以匹配包含有单独的单词 a 的字符串
/a\B/        //可以匹配任意包含字符 a 的字符串。如果只含一个 a，则 a 不能是结尾字符
```

7.5.4 Perl 正则表达式函数

处理 Perl 正则表达式的函数主要有 4 个，分别是 preg_match()、preg_match_all()、preg_replace()和 preg_replace_callback()。

1. preg_match()函数

preg_match()函数用于从目标字符串中搜索与指定正则表达式相匹配的内容。函数语法格式如下：

```
int preg_match ( string $pattern , string $subject [, array &$matches [,
int $flags = 0 [, int $offset = 0 ]]] )
```

preg_match()函数返回$pattern 所匹配的次数。不过该函数在第一次匹配之后将停止搜索，所以如果没有匹配，则返回 0，否则返回 1。

【实践案例 7-22】

下面的代码演示了如何使用 preg_match()函数匹配一个字符串、一个单词、一个日期以及一个数字。

```
<?php
$str="God is A Girl,you are my super star";//匹配字符串
//模式分隔符后的"i"标记表示不区分大小写的搜索方式
echo "匹配 gi 字符串结果：";
if (preg_match("/gi/i", $str)) {
    echo "匹配成功。";
} else {
    echo "匹配失败。";
}
echo "\n\n";
/* 模式中的\b 标记一个单词边界，所以只有独立的单词"re"会被匹配*/
echo "匹配 re 单词结果：";                    //匹配单词
if (preg_match("/\bre\b/i", $str)) {
    echo "匹配成功。";
} else {
    echo "匹配失败。";
```

```
}
echo "\n\n";

$dateStr="今天是 2012-08-08";                           //匹配日期
echo "匹配日期的结果：";
//从 URL 中获取主机名称
preg_match('#\d{4}-\d{1,2}-\d{1,2}#',$dateStr, $matches);
echo $matches[0];

$date = $matches[0];                                     //匹配数字
preg_match('#\d+#', $date, $matches);
echo "\n 年份是： {$matches[0]}";
?>
```

上述代码执行后将看到如下输出结果：

```
匹配 gi 字符串结果：匹配成功。

匹配 re 单词结果：匹配失败。

匹配日期的结果：2012-08-08
年份是： 2012
```

2. preg_match_all()函数

preg_match_all()函数的作用与 preg_match()函数差不多。不同的是，preg_match_all()函数会一直搜索到目标字符串的结尾处。preg_match_all()函数的语法结构如下：

```
int preg_match_all ( string $pattern , string $subject , array &$matches
[, int $flags = PREG_PATTERN_ORDER [, int $offset = 0 ]] )
```

该函数在执行时将搜索$subject 中所有匹配$pattern 给定正则表达式的匹配结果，并且将它们以$flag 指定顺序输出到$matches 中。最终返回完整匹配次数（可能是 0），如果发生错误则返回 false。

【实践案例 7-23】

下面的代码演示了如何使用 preg_match_all()函数查找匹配的日期：

```
<?php
$str = "今天是 2012-08-08，明天 2012-8-9 是我的生日";
preg_match_all("#(\d{4})-(\d{1,2})-(\d{1,2})#", $str, $matches, PREG_
SET_ORDER);
print_r($matches);
?>
```

上述代码执行后输出结果如下：

```
Array
```

```
(
    [0] => Array
        (
            [0] => 2012-08-08
            [1] => 2012
            [2] => 08
            [3] => 08
        )
    [1] => Array
        (
            [0] => 2012-8-9
            [1] => 2012
            [2] => 8
            [3] => 9
        )
)
```

3. preg_replace()函数

preg_replace()函数的语法格式如下所示。

```
mixed preg replace ( mixed $pattern , mixed $replacement , mixed $subject
[, int $limit = -1 [, int &$count ]] )
```

它将搜索$subject中匹配$pattern的部分，并使用$replacement进行替换。可选的$limit用于限定匹配次数，如果指定了$count，将会被填充为完成的替换次数。

【实践案例7-24】

preg_replace()函数可以快速地对多个关键字进行替换。例如，要对用户输入的内容进行过滤，替换不文明的用语，使用preg_replace()函数实现代码如下：

```
<?php
$words="猪头，这么简单都不会，真是笨死了。你真垃圾，去死吧！";  //要过滤的字符串
$badwords=array("/猪头/","/垃圾/","/去死/");                //要替换的关键字
$goodwords=array("宝贝","可爱","一边玩去");                  //替换的内容
echo "原文：".$words;
echo "\n 替换：".preg_replace($badwords, $goodwords,$words);
?>
```

上述代码会将在$words中出现$badwords定义的关键字替换为$goodwords中对应的内容。执行后将看到如下输出结果：

```
原文：猪头，这么简单都不会，真是笨死了。你真垃圾，去死吧！
替换：宝贝，这么简单都不会，真是笨死了。你真可爱，一边玩去吧！
```

4. preg_replace_callback()函数

preg_replace_callback()函数可以实现调用指定函数来处理字符串替换操作，函数的语

法格式如下：

```
mixed preg_replace_callback ( mixed $pattern , callback $callback , mixed
$subject [, int $limit = -1 [, int &$count ]] )
```

与 preg_replace()函数相比，preg_replace_callback()函数除了可以指定一个$callback 替代 preg_replace()函数中的$replacement 进行替换字符串的计算之外，其他方面都完全相同。

【实践案例 7-25】

preg_replace_callback()函数在实际工作中非常有用，例如，对论坛、博客或者新闻中评论的非法言论和文字进行过滤。下面的代码演示了该函数实现字符串替换操作的代码：

```
<?php
function translate($match){          //自定义的正则表达式替换函数
    $arr = array(                    //定义替换列表
            ':face:' => '<img src="smile.jpg">',
                                     //键为匹配的关键字,值为替换的内容
            ':qq:' => '12345678',
            ':homepage:' => 'www.itzcn.com',
            ':hello:'=>"你好呀"
    );
    if(isset($arr[$match[0]])){      //如果找到匹配的，则使用列表中对应的值
        return $arr[$match[0]];
    }else {                          //没有则使用原始内容
        return $match[0];
    }
}
$words=":hello:，我的朋友:face:，我的个人网站:homepage:，QQ:qq:";
$result = preg replace callback("/:\w+:/", 'translate', $words);
echo $result;
?>
```

上述代码在 preg_replace_callback()函数中调用 translate()函数进行字符串替换处理，在 translate()函数中将传递过来的字符串与数组 arr 中的键进行比较，如果存在此键，则获得对应的值，并替换后输出。执行后将看到如下输出结果：

```
你好呀，我的朋友 ，我的个人网站www.itzcn.com，QQ12345678
```

7.6　项目案例：实现基于 XML 的广告位管理

在很多网站系统中，为了使网站可以更好地运营，都提供了很多的广告位。通过广告位允许其他站长发布广告和推广内容。这就需要保存广告位的类型、投放位置、尺寸以及发布时间等信息。为了方便修改，这些广告信息通常都保存在网站外部的 XML 文件中。下面就使用本章所学的知识实现用 XML 保存的广告位管理功能。

【实例分析】

首先需要创建一个 XML 文件，使用节点来定义要存储的广告位信息。本实例中最终采用的 XML 文档结构如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<ads>
  <title>taotao 商城内部广告位管理系统</title>
  <updatedate>2012-08-14 15:53:31</updatedate>
  <userad>
    <id>广告编号</id>
    <username>发布者名称</username>
    <sitename>网站名称</sitename>
    <status>广告状态（禁用、启用）</status>
    <type>广告类型（文本、图片）</type>
    <targetsite>投放位置</targetsite>
    <size>广告尺寸</size>
    <onlinedate>发布时间</onlinedate>
  </userad>
</ads>
```

按照上述结构以 adSetting.xml 为文件名进行保存。然后实现添加广告位、显示广告位列表和删除广告的功能，具体步骤如下。

（1）在 adSetting.xml 所在目录新建一个 index.php 文件。

（2）在 index.php 的顶部添加如下的代码来定义 getmicrotime()函数：

```
<?php
function getmicrotime()
{
list($usec, $sec) = explode(" ",microtime());
return ((float)$usec + (float)$sec);
}
?>
```

getmicrotime()函数用于获取一个毫秒时间戳。这个函数将在后面调用，所以在这里先进行了定义。

（3）为了计算页面执行时间，需要在页面开始时获取一下时间，在页面结束时再获取一下时间，然后计算两个时间的差。这里在 index.php 文件顶部添加如下代码获取第一个时间：

```
<? php
$time_start = getmicrotime();       //获取页面开始加载时时间
?>
```

（4）在页面的最后，添加如下的代码获取第二个时间，并计算时间差显示执行时间。具体实现代码如下所示：

```
<? php
$time_end = getmicrotime();                    //获取页面结束后时间
printf ("[页面执行时间: %.2f 毫秒]\n\n",($time_end - $time_start)*1000);
                                               //计算时间差
?>
```


（5）在 getmicrotime()函数下方创建一个 addAdData()函数。addAdData()函数的主要作用是保存用户输入的新广告位信息，即向 adSetting.xml 中新建一个 userad 节点。具体实现代码如下所示：

```
function addAdData($name,$sitename,$status,$type,$size,$target)
                                       //添加广告位函数
{
    $xml = simplexml load file("adSetting.xml");
    $userad=$xml->addChild("userad");
    $i=count($xml->xpath("/ads/userad"))+1;
    $userad->addChild("id",$i);
    $userad->addChild("username",$name);
    $userad->addChild("sitename",$sitename);
    $userad->addChild("status",$status);
    $userad->addChild("type",$type);
    $userad->addChild("size",$size);
    $userad->addChild("targetsite",$target);
    $userad->addChild("onlinedate",date("Y-m-d H:i:s"));
    $xml->asXML("adSetting.xml");                  //保存到文件
    echo "<script>alert('添加成功');</script>";      //弹出提示
}
```

在这里使用 SimpleXML 方式来实现添加。addChild()函数用于添加一个节点，它有两种形式，带有一个参数时将仅仅创建节点，不填充数据；使用两个参数时可以同时指定节点名称和数据。xpath()函数可以执行一个 XML 的查询，并返回一个符合条件的数组，在这里使用它根据已有 userad 节点的数量生成新的编号。最后的 asXML()函数可以将修改保存到一个文件。

（6）创建一个用于删除广告位的 delAdData()函数，在删除时使用广告编号作为标识。具体实现代码如下：

```
function delAdData($id)                        //$id 参数为要删除的广告编号
{
    $dom = new DOMDocument("1.0");
    $dom->load('adSetting.xml');
    $ads=$dom->documentElement->getElementsByTagName('userad');
    foreach($ads as $ad)
    {
        $idNode = $ad->getElementsByTagName("id");
                                               //获取指定节点对象的节点名值
```

```
            $idName = $idNode->item(0)->nodeValue;
            if($idName==$id)                    //如果找到要删除的广告编号
            {
                $ad->parentNode->removeChild($ad); //通过父节点删除当前节点
            }
        }
        $dom->save("adSetting.xml");
    }
```

如上述代码所示，在这里使用 DOM 方式实现删除节点。首先获取所有的 userad 节点，然后遍历并判断其中 id 子节点的值是否为要删除的 id，如果相同则通过父节点的删除方法进行删除，最后再将修改保存到原文件中。

（7）除了添加和删除之外，还需要一个列表显示当前所有的广告信息。这个功能通过 LoadAdData()函数实现，如下所示是该函数的代码：

```
     <?php
       function LoadAdData(){                   //显示广告位列表函数
         $xml = simplexml_load_file("adSetting.xml");
   ?>
         <table class="listing" cellpadding="0" cellspacing="0">
           <tr>
             <th class="first" width="39">编号</th>
             <th width="100">用户名称</th>
             <th width="135">站点名称</th>
             <th width="68">显示位置</th>
             <th width="39">状态</th>
             <th width="45">类型</th>
             <th width="42">尺寸</th>
             <th width="103">上线时间</th>
             <th width="40" class="last">操作</th>
           </tr>
           <?php
           $rows=1;
           $ads=$xml->xpath("/ads/userad");           //选择所有 userad 节点
           foreach($ads as $ad)                       //遍历 userad 节点
           {
               $style=$rows%2==0?"":"bg";
               $rows++;
               $status=$ad->status=="0"?"禁用":"启用";
               $type=$ad->type=="text"?"文本":"图片";
           ?>
           <tr class="<?php echo $style;?>">
           <td class="first style1"><?php echo $ad->id;?></td>
           <td><?php echo $ad->username;?></td>
           <td><?php echo $ad->sitename;?></td>
```

```
        <td><?php echo $ad->targetsite;?></td>
        <td><?php echo $status;?></td>
        <td><?php echo $type;?></td>
        <td><?php echo $ad->size;?></td>
        <td><?php echo $ad->onlinedate;?></td>
        <td class="last"><a href="?del=<?php echo $ad->id;?>"><img src=
       "img/hr.gif" width="16" height="16" alt="" /></a></td>
        </tr>
        <?php }?>
      </table>
   <?php }?>
```

（8）经过上面的步骤，实例所需的函数就编写完成了。下面在 index.php 文件的重要位置读取 XML 文件中的站点名称和更新时间，并在合适位置调用 LoadAdData()函数显示广告列表。在实例中使用的代码如下所示：

```
<?php
$xml = simplexml_load_file("adSetting.xml");        //载入 XM
echo $xml->title;
echo "    上次更新时间: ".$xml->updatedate;
$xml=null;
?>
</h3><div class="table"> <img src="img/bg-th-left.gif" width="8" heig"
alt="" class="left" /> <img src="img/bg-th-right.gif" width="7" heig" alt=""
class="right" />
 <?php LoadAdData();?>     <!-- 调用 LoadAdData()函数显示广告列表      -->
</div>
```

（9）在列表的下方使用 form 表单制作广告位的添加布局，代码如下所示：

```
<form id="form1" name="form1" method="post" action="index.php">
  <table width="90%" border="0" align="center" cellpadding="2" cellcing=
  "2">
    <tr>
      <td>会员名称:
      <input name="username" type="text" id="username" size="20" /></td>
      <td>类型:     <br />   <label><input type="radio" name="type" valutext"
      id="RadioGroup2 0" />文本</label>
         <label><input type="radio" name="type" value="images" id=
         "ioGroup2_1" /> 图片</label>         </td>
             <td>状态:  <br /><label><input type="radio" name="status" valu0"
             id="RadioGroup1_0" /> 禁用</label>
               <label> <input type="radio" name="status" value="1"
               id="ioGroup1_1" />启用</label>
      </td>
          </tr>
```

```
            <tr><td>网站名称:  <input name="sitename" type="text" id="ename"
            size="20" /></td>
              <td>显示位置:  <input name="target" type="text" id="target" size0"
              /></td>
              <td>广告尺寸:  <input name="size" type="text" id="size" size0" />
              </td>
            </tr>
            <tr>
              <td colspan="3"><input type="submit" name="button" id="ton"
              value="确定" /></td>
            </tr>
          </table>
        </form>
```

在上面的表单中根据 XML 的文档结构分别制作用于输入信息文本框。在这里要注意 input 标记和“确定”按钮的 name 属性。

（10）现在运行实例，由于 XML 文档为空所以会看到空列表。在列表下面是添加表单，可以输入内容后单击“确定”按钮进行添加。页面最底显示了页面的执行时间，如图 7-8 所示。

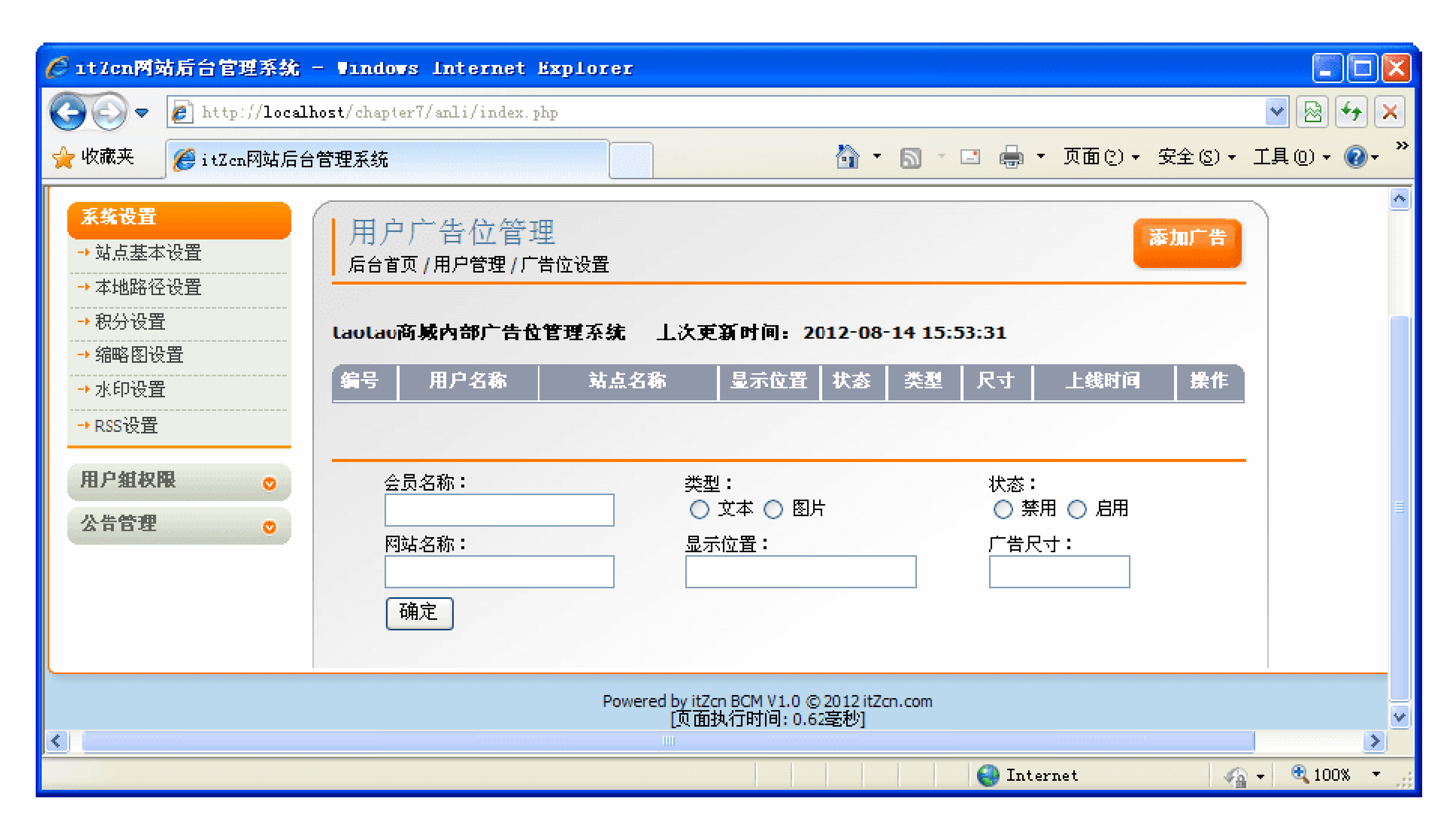


图 7-8　首次运行效果

（11）接下来编写单击“确定”按钮之后的添加代码，以及用于删除的代码。

```
<?php
if(isset($ POST["button"]))                          //单击“确定”按钮
{
    $username=$_POST['username'];
    $sitename=$ POST['sitename'];
    $status=$ POST['status'];
    $type=$ POST['type'];
    $target=$_POST['target'];
```

```
        $size=$ POST['size'];
        addAdData($username,$sitename,$status,$type,$size,$target);
                                                        //调用添加函数
    }
    if(isset($ GET["del"]))                             //单击“删除”链接
    {
        $delid=$_GET["del"];
        delAdData($delid);                              //调用删除函数
    }
    ?>
```

（12）再次运行然后使用添加功能新建一些广告数据，还可以单击“操作”列中的图标进行删除。图 7-9 所示为显示 3 条广告位的列表效果。

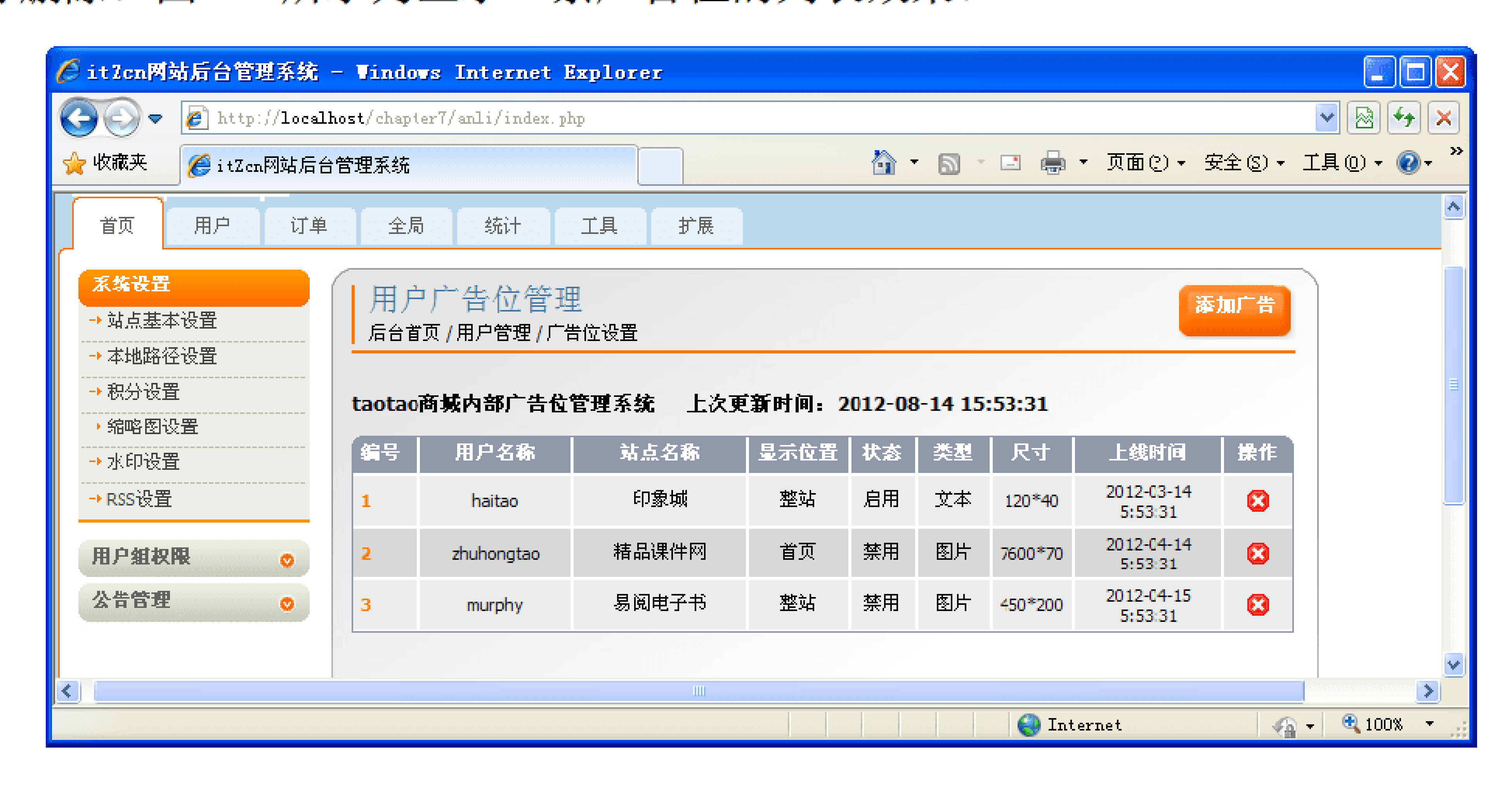


图 7-9　显示广告位列表

7.7　习题

一、填空题

（1）__________是 PHP 默认的参数传递方式，它会为函数外部变量的值创建一个副本，然后赋给函数内部的局部变量。

（2）下面代码创建的 SumArgs()函数可以对调用时传递的所有参数进行求和，并输出结果。在空白处填写合适的代码，使函数可以正确运行。

```
    function SumArgs(){
        $result=0;                                          //初始化返回结果
        $num= __________;                                   //获取参数数量
        echo "参数的个数为: $num<br>";
        $test= __________;                                  //获取参数数组
        for($i=1;$i<$num+1;$i++){
```

```
        printf("第%s 个参数是: %s <br>",$i,$test[$i-1]);
        $result+=$test[$i-1];                          //执行累加计算
    }
    echo "所有参数之和为: $result ";
}
```


（3）要产生一个 1 到 100 之间的随机数，应该使用代码__________。

（4）下面代码使用递归函数实现求一个数的阶乘，实现代码如下所示。在空白处填写合适的代码，使函数可以正确运行。

```
<?php
function factorial($number)             //创建函数
{
    $result=1;
    if($number<=1)
    {
        $result=1;                      //返回函数执行结果
    }
    else
    {
       $result=$number* __________;     //递归调用函数本身
    }
    return $result;                     //返回函数执行结果
}
?>
```

（5）假设$str 变量的值是“a12$34”，下面代码运行后的输出是__________。

```
<?php
$str = '';
if (preg match("^[a-zA-Z0-9]{4,16}$^", $str)) {
   echo "成功";
} else {
   echo "失败";
}
?>
```

二、选择题

（1）下面关于自定义函数的描述，错误的是__________。

A．函数名称区分大小写

B．使用 return 语句指定返回值

C．参数数量不限

D．函数体可以为空

（2）下列选项中不属于 PHP 参数传递方式的是__________。

A．按值传递

B．按引用传递

C．按地址传递

D．按默认值传递

（3）假设有如下的 PHP 代码，执行后的输出结果为__________。

```
<?php
function test(){
    return func num args();
}
echo test('a',array("b","c"),"d",123);
?>
```

A．1

B．2

C．3

D．4

（4）getdate()函数返回的数组中，__________键表示 4 位数字的完整年份。

A．year

B．yday

C．y4

D．y

（5）使用 date()函数时要格式化为“XXXX 年 XX 月 XX 日”的日期，应该使用格式字符串__________。

A．Y 年 n 月 j 日

B．Y 年 M 月 D 日

C．yyyy 年 mm 月 dd 日

D．year 年 mon 月 mday 日

（6）在 POSIX 风格的正则表达式中使用__________匹配所有大写字母。

A．[:alnum:]

B．[:alpha:]

C．[:digit:]

D．[:upper:]

三、上机练习

1．制作一个七星彩的预测程序

利用 PHP 随机函数制作一个七星彩的预测程序，并且要求结果以图片的形式显示。七星彩是一个七位的数字，每一位都是 0~9 之间的随机数，也就是说它的范围是 0000000~9999999。

执行后的效果如图 7-10 所示。

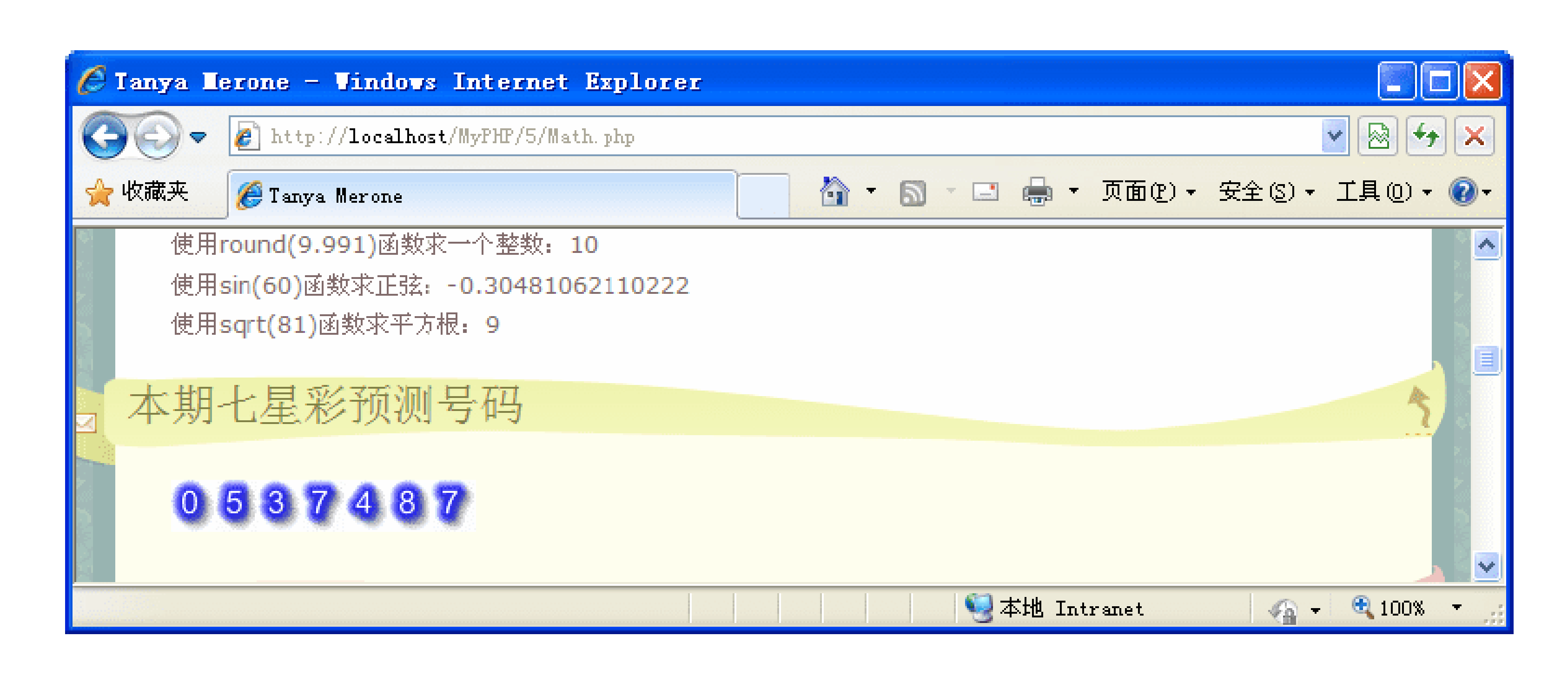


图 7-10　随机数运行效果

2. 操作学生信息 XML 文档

假设，有一个保存学生信息的 XML 文档结构如下所示：

```
<students>
    <student id="学号">
        <name>姓名</name>
        <address>地址</address>
        <age>年龄</age>
    </student>
</students>
```

根据上面的 XML 文档结构实现学生信息的列表显示、增加、删除和修改。

实现思路：一种比较简单的方法是通过 DOM 来实现。因为 DOM 不仅可以实现解析 XML 文档，还可以建立 XML 文档、为 XML 文档修改节点、增加 XML 文档节点和删除 XML 文档节点。

7.8　实践疑难解答

7.8.1　使用 date()函数出错的问题



使用 date()函数出错的问题

网络课堂：http://bbs.itzcn.com/thread-19689-1-1.html

【问题描述】：data()本来是一个非常简单的 PHP 日期函数，以前也经常使用。谁知道今天使用 date()函数时出错了。错误提示如下：

```
date() expects parameter 2 to be long, string given...
```

如果在前面加@又能正常使用。这是为什么呢？

【解决办法】：

从报出的错误来看是你 date()的参数设置不正确。

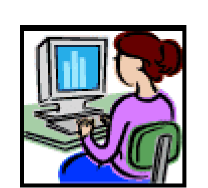
错误提示是：date() expects parameter 2 to be long, string given

错误提示的中文意思是：date 函数需要参数 2 为 long 类型,已赋值的类型为 string 类型例如，下面的代码：

```
<?=@date("Y-m-d H:i:s","$row[subtime]");?>
```


这里第二个参数$row[subtime]虽然是一个类似于 1339746163 的时间戳，但是由于有引号，会作为字符串。这就能看出问题了吧，解决办法是去掉引号。如果还是有问题的话，就代表$row[subtime]是个字符串类型，解决很简单，后面加个 0 就转换了。

7.8.2　SimpleXML 的一点注意事项



SimpleXML 的一点注意事项
网络课堂：http://bbs.itzcn.com/thread-19690-1-1.html

【问题描述】：

关于 PHP 中使用 SimepleXML 操作 XML 时需要注意的内容有哪些？

这是面试时的一道题目，请大家给总结一下都有哪些。

【正确回答】：SimpleXML 提供了一套简单快速的 XML 操作方法，大大地提高了 XML 操作的效率，但是有时不小心也会带来不小的麻烦，看下面一段代码：

```
$xml = simplexml_load_string('<root><title>title</title></root>');
$title = $xml->title;
echo $title;
$xml->title = 'test';
echo $title;
```

猜猜第二个输出结果会是多少？是'test'，而不是想象中的'title'。为什么会这样呢？原因在这里：

```
echo gettype($xml->title)            //输出: object
echo get_class($xml->title);         //输出: SimpleXMLElement
```

看到了吗，$xml->title 是一个 SimpleXMLElement 类的实例，而不是字符串。所以$title 实际上保存的是一个到 SimpleXMLElement 类的一个引用，而不是字符串的副本。想要得到字符串的副本只能进行类型转换：

```
$title = (string)$xml->title;        //获得字符串
$xml->title = 'test';
echo $title;                     //输出 title
```

7.8.3　请教 PHP 正则表达式过滤和替换的问题



请教 PHP 正则表达式过滤和替换的问题
网络课堂：http://bbs.itzcn.com/thread-19691-1-1.html

【问题描述】：没有其他语言基础，最近在学 PHP，遇到一个正则表达式的问题。请教

大家给个解决办法或者思路。

假设有如下的一段 HTML 内容：

```
<P>内容 1</P> <p id=pp123>内容 2</p> <p class=123>内容 3</p>
```

要实现的是先过滤 P 里所有的 id 和 class 属性，然后根据 p 标记的数量按顺序给 p 加上 id 属性。最终通过正则期望得到下面结果：

```
<P id=1>内容 1</P>
<P id=2>内容 2</P>
<P id=3>内容 3</P>
```

【解决办法】：这个功能使用普通的正则表达式替换函数实现不了。必须自定义匹配后的替换函数，也就是自定义回调函数，大致方法如下。

（1）使用 preg_replace("/<p(.*?)>/","",$str)函数进行替换。

（2）替换完后使用 explode("</p>",$str)函数再拆分成数组。

（3）再循环处理数组加 id 进去。

当然，可以直接用 preg_replace_callback()函数的处理方式一步到位。测试代码如下：

```
<?php
$content = '<P>内容 1</P> <p ID=pp123>内容 2</p> <p class=123>内容 3</p>';
$parten  = '/<p[^>]*>/i';
$new = preg replace callback($parten,'translate',$content);
                                    //调用 preg_replace_callback()函数
echo $new;
function translate($match){
    return getNewId($match);
}
function getNewId($matches) {
    static $i = 1;
    $html = "<p id=$i>";
    $i++;
    return $html;
}
?>
```

第8章 文件和目录处理

在网站程序中经常需要访问保存在本地文件系统中的数据，像查看配置文件是否存在、读取文件内容、向文件中写入新内容、重命名文件以及遍历目录等。在 Web 编程中，文件的操作一直是开发人员比较棘手的问题，而 PHP 提供了一套强大的函数处理文件和目录。

本章将详细讨论它们，像获取文件的大小、读取文件的一行、写入内容、删除文件、创建目录、解析文件名以及获取可用空间等。通过本章的学习，读者将能够很方便地操作文件系统上的文件和目录。

本章学习要点：

- 掌握获取文件各种属性的方法
- 掌握如何打开和关闭文件
- 掌握按行和按字节等读取函数的使用
- 了解文件指针的应用
- 熟悉复制、重命名和删除文件的操作
- 掌握打开和关闭目录的方法
- 掌握如何遍历目录
- 熟悉解析路径的各种方法
- 掌握获取磁盘容量的方法

8.1 查看文件属性

在计算机系统中，数据是以文件的形式被保存到磁盘中的。为了容易区分不同的数据，就需要尽可能多的对文件进行描述，这个描述就是文件的属性。例如，文本文件的内容、音乐文件的时间长度、图片文件的大小和数据库文件的创建时间等。

在 PHP 中提供了很多系统函数用于获取文件的各种属性，表 8-1 列出了常用的函数及说明。

表 8-1　获取文件属性函数

函数名称	作用说明
filetype()	获取指定文件或目录的类型
filesize()	获取指定文件的大小，以字节为单位
fileinode()	获取文件的 inode 编号

续表

函数名称	作用说明
fileatime()	获取文件上次被访问的时间戳
filectime()	获取文件最后一次被修改的时间戳
filemtime()	获取文件上次被修改的时间戳
filegroup()	以数字格式获取指定文件的组 ID
fileowner()	以数字格式获取文件所有的用户 ID
fileperms()	获取文件的访问权限
is_executable()	用于判断指定文件是否可执行
is_readable()	用于判断指定文件是否可读
fstat()	以数组形式获取指定文件的属性信息

8.1.1 filetype()函数

filetype()函数可以获取指定文件的类型，它的语法格式如下所示：

```
string filetype ( string $filename )
```

它执行成功将会返回一个表示$filename 文件类型的标识符。该标识符有 7 个可能的值，如下所示。

- **Block：表示驱动器或 CD-ROM。**
- **char：表示字符设备。**
- **dir：表示目录。**
- **fifo：表示命名管道。**
- **file：表示硬链接。**
- **link：表示符号链接。**
- **unknown：表示未知类型。**

filetype()函数执行失败则返回 false。如果 stat 调用失败或者文件类型未知的话，filetype()还会产生一个 E_NOTICE 消息。

【实践案例 8-1】

假设在网站 doc 目录有一个 help.doc 文件，现在要显示该文件的一些信息，像文件大小、最近访问时间和文件类型等。

实现这个功能就需要使用表 8-1 中介绍的各种属性获取函数，具体实现代码如下：

```
<p class="h1">文档【help.doc】属性</p>
  <ul class="t1">
    <?php
        $filename = "doc/help.doc";                              //定义文件名
        echo "<li>文件大小为: " . filesize ( $filename ) . "bytes</li>";
        echo "<li>最近访问时间: ".date("FdYH:i:s",fileatime($filename)).
        "</li>";
        echo "<li>最近权限修改时间: " . date ( "F d Y H:i:s", filectime
        ( $filename) ) . "</li>";
```

```
        echo "<li>上次内容修改时间: " . date ( "F d Y H:i:s", filemtime
        ( $filename ) ) . "</li>";
        echo "<li>文件所有者: " . fileowner ( $filename ) . "</li>";
        echo "<li>文件类型: " . filetype ( $filename ) . "</li>";
    ?>
</ul>
```

保存上述代码为 index.php，运行后将看到如图 8-1 所示的效果。

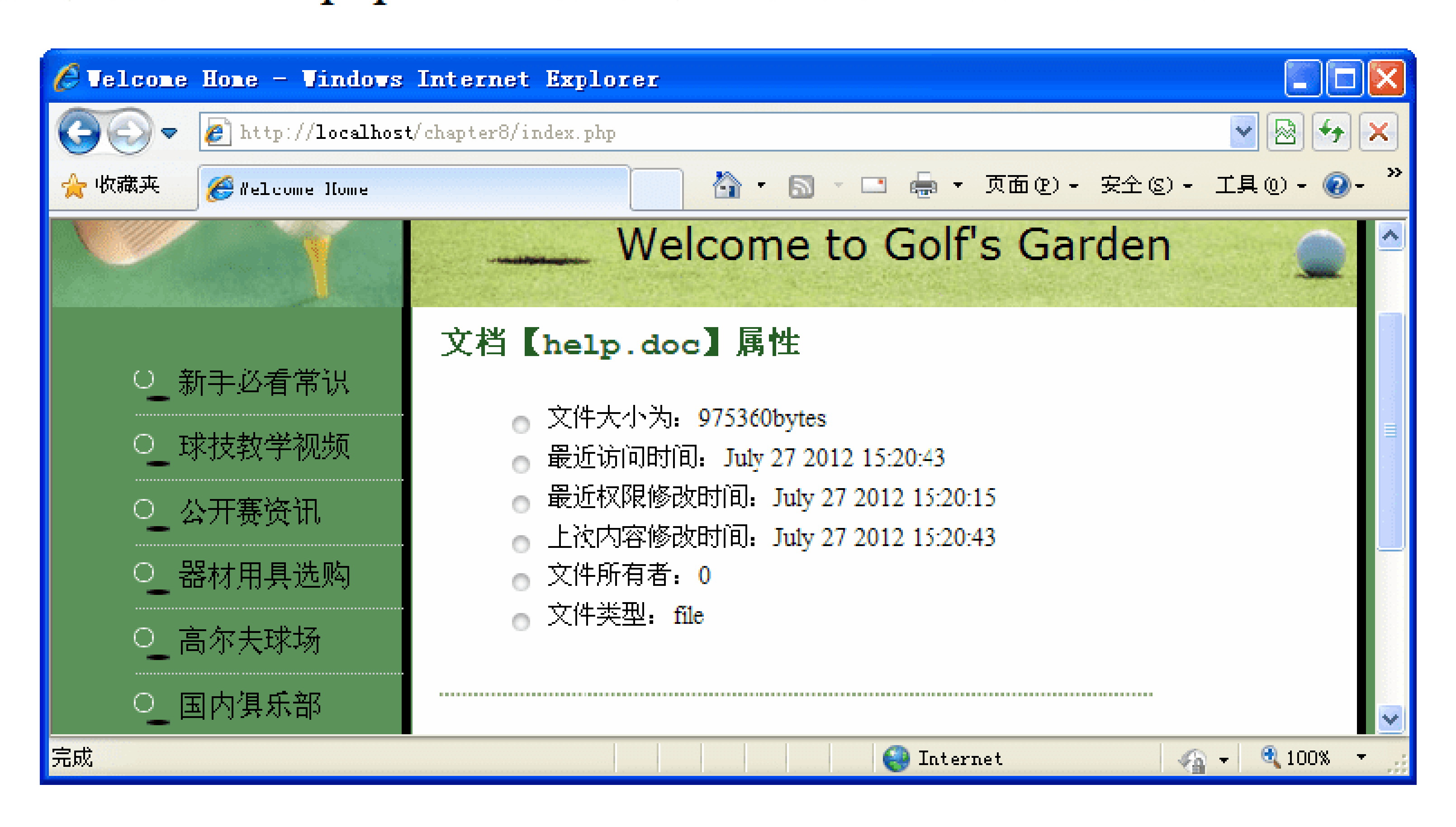


图 8-1　查看文件属性效果

8.1.2　fstat()函数

使用 fstat()函数同样可以获取文件的各种属性，与其他函数不同的是它要求文件必须已打开，才能获取文件信息。语法格式如下：

```
array fstat ( resource $handle )
```

这里的$handle 就是一个文件打开的指针，执行后返回一个包括文件信息的数组。

下面代码演示如何使用 fsta()函数读取 doc/help.doc 文件的属性。

```
<?php
$fp = fopen("doc/help.doc", "r");          //打开文件
$fstat = fstat($fp);                       //取得统计信息
fclose($fp);                               //关闭文件
print_r(array_slice($fstat, 13));          //只显示关联数组部分
?>
```

上述代码首先打开一个到 doc/help.doc 文件的指针$fp，再使用 fstat($fp)将指针所引用文件的属性信息保存到$fstat 数组中。由于$fstat 数组中包含了很多重复值，在这里只显示数组部分，输出结果如下所示：

```
Array
(
```

```
    [dev] => 0
    [ino] => 0
    [mode] => 33206
    [nlink] => 1
    [uid] => 0
    [gid] => 0
    [rdev] => 0
    [size] => 975360
    [atime] => 1343373643
    [mtime] => 1343373643
    [ctime] => 1343373615
    [blksize] => -1
    [blocks] => -1
)
```

8.2 打开和关闭文件

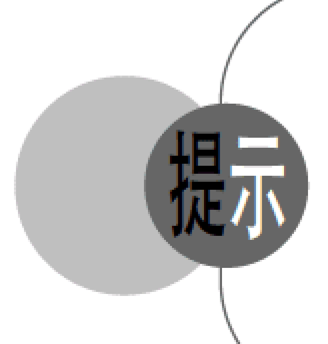
在对文件进行操作之前都必须先打开再操作，而在操作结束时，必须执行关闭操作。打开文件的目的是创建一个文件的缓冲区，以方便后面进行输入/输出操作，而关闭文件的目的则是将缓冲区中未写入的数据保存到文件。

8.2.1 打开文件

fopen()函数可以打开本地和 URL 上的文件，函数语法格式如下：

```
resource fopen ( string $filename , string $mode [, bool $use_include_path
[, resource $zcontext ]] )
```

fopen()函数会将$filename 指定的文件资源绑定到一个流或句柄上。绑定之后，就可以通过该句柄与文件进行交互。



如果$filename 指定的是一个本地文件，PHP 将尝试在文件上打开一个流。该文件必须是 PHP 可以访问的，因此需要确认文件访问权限允许访问。如果指定的是一个已注册的协议，而协议被注册为一个网络 URL，PHP 将检查并确认 allow_url_fopen 已被激活。

mode 参数表示文件打开方式，主要用于确定用户访问资源的模式，在表 8-2 中列出了 mode 参数的可选值。

表 8-2　mode 参数模式

mode	说明
"r"	只读方式打开，将文件指针指向文件头

续表

mode	说明
"r+"	读写方式打开，将文件指针指向文件头
"w"	写入方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建
"w+"	读写方式打开，将文件指针指向文件头并将文件大小截为零。如果文件不存在则尝试创建
"a"	写入方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建一个新文件
"a+"	读写方式打开，将文件指针指向文件末尾。如果文件不存在则尝试创建新文件
"x"	创建并以写入方式打开，将文件指针指向文件头。如果文件已存在，则 fopen()调用失败并返回 false，并生成一条 E_WARNING 级别的错误信息
"x+"	创建并以读写方式打开，将文件指针指向文件头。如果文件已存在，则 fopen()调用失败并返回 false，并生成一条 E_WARNING 级别的错误信息

如果资源位于本地文件，PHP 则认为可以使用本地路径或相对路径来访问此资源。或者可以将 fopen()的$use_include_path 参数设置为 true，这样会使 PHP 考虑配置指令 include_path 中指定的路径。参数$zcontent 是用来设置文件或流特有的配置参数，以及在多个 fopen()请求之间共享文件或流特有的信息。如果打开失败，函数将返回 false。

例如，下面代码演示如何使用 fopen()函数以只读方式打开、读写方式打开、打开本地和网络文件等。

```
<?php
$file = fopen("musicList.txt","r");                          //只读方式打开
$file = fopen ("uploadFile/doc/en.doc", "r+");               //读写方式打开
$file = fopen ("siteConfig/users.xml","wb");                 //写入
$file = fopen ("http://www.itzcn.com/videos.html", "r");     //打开网络文件
$file = fopen ("ftp://user:password@itzcn.com/videos.html", "w");
                                                     //使用 ftp 协议
$file = fopen("c:\\windows\\system\\boot.inf","r");
                                                     //使用绝对路径打开本地文件
?>
```



在使用绝对路径指定文件时，要注意转义文件路径中的每个反斜线，或者用斜线。

8.2.2 关闭文件

在 PHP 中使用 fclose()函数关闭已经打开的文件，该函数的语法格式如下所示：

```
bool fclose ( resource $handle)
```

fclose()函数将关闭之前打开的由$handle 指定的文件句柄，如果成功则返回 true，否则返回 false。这里要注意，参数$handle 必须是 fopen()或者 fsockopen()成功打开的文件句柄。

下面代码演示如何使用 fclose()函数关闭由 fopen()打开的文件：

```
<?php
```

```
  $handle = fopen('doc/help.doc', 'r');//使用$handle打开文件doc/help.doc
  fclose($handle);                     //关闭$handle
?>
```


8.3 读取文件

文件打开之后，便可以对文件的内容进行各种操作。本节首先学习如何读取文件，像读取一行、读取全部或者读取指定字节等。

8.3.1 读取一行

PHP 提供了 4 个函数用于从文件中读取一行内容，分别是 file()、fgets()、fgetss()和 fgetcsv()。

1. file()函数

使用 file()函数可以将整个文件读入到一个数组中，该函数的语法格式如下：

```
array file(string $filename[, int $use_include_path [, $resource context]])
```

其中各个参数含义如下。

- **$filename　指定要读取的文件路径。**
- **$use_include_path　如果需要在**$use_include_path **路径中搜寻文件，可以将此参数设为** 1。
- **$context　可选。指定文件句柄的环境，若使用** null **则忽略。**

file()函数执行成功返回一个数组，数组中的每个元素都对应文件中的一行，包括换行符在内。如果执行失败则返回 false。

【实践案例 8-2】

在 doc/game.txt 文件中保存了有关网站的资讯内容，现在需要在页面上显示出来，并在显示时添加上行号。

使用 file()函数的实现代码如下所示：

```
<p class="h1">查看赛事内容</p>
  <ul class="t1">
  <?php
    $filename = "doc/game.txt";          //定义文件名
    $lines = file($filename);            //将一个文件读入数组$lines
    //遍历数组，为文件中的内容加上行号
    foreach ($lines as $line num => $line) {
        echo "<li>NO #<b>{$line num}</b> : " .$line . "</li>";
    }
    ?>
```

```
    </ul>
```

上述代码使用 file()读取 doc/game.txt 文件的内容，并将内容以数组形式保存到$lines 中。由于 file()函数是逐行读取的，所以可以在遍历时添加行号。最后运行后的效果如图 8-2 所示。

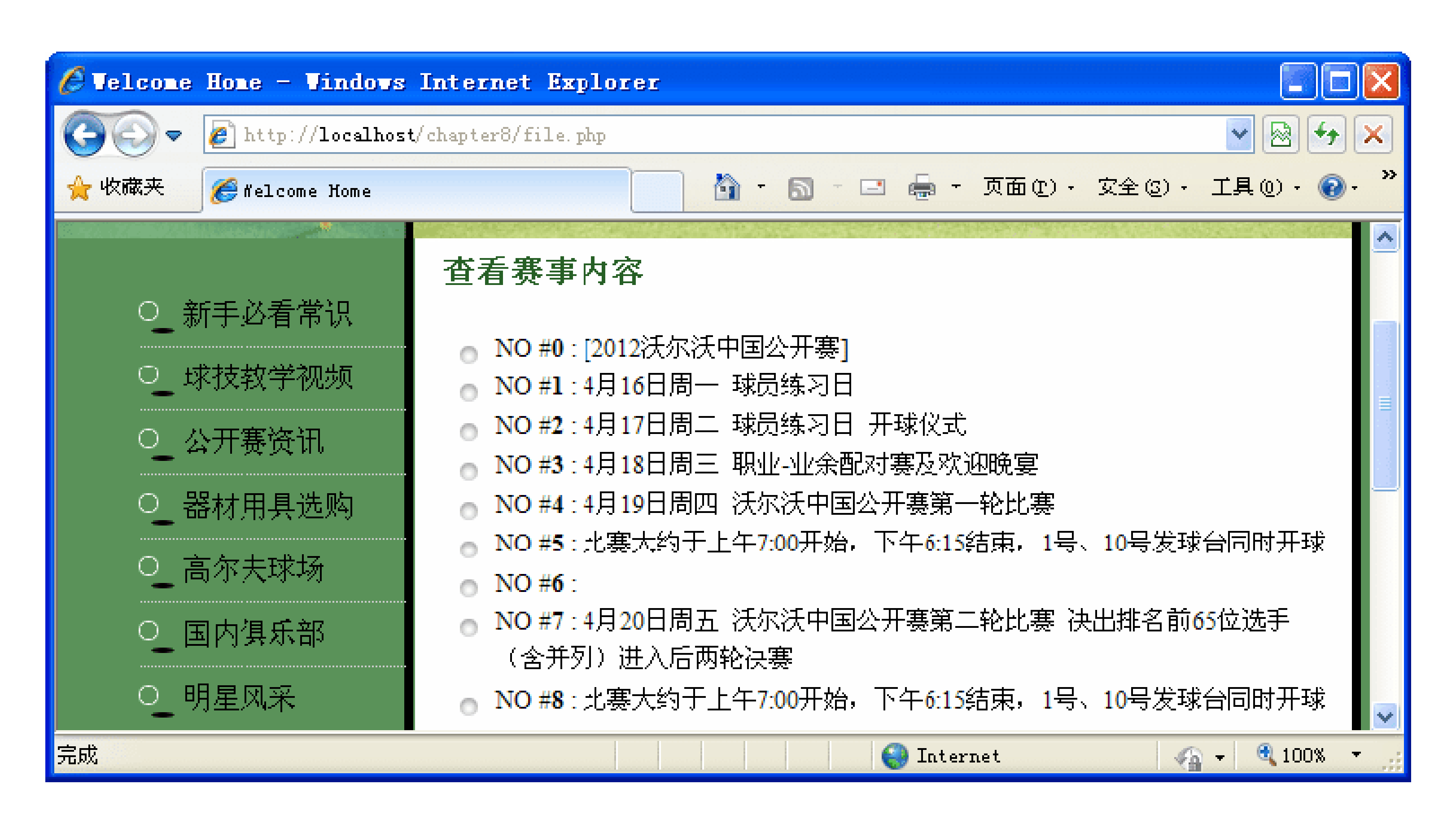
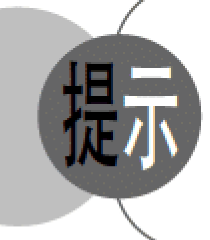


图 8-2　使用 file()函数读取文件



file()函数返回的数组中每一行都包括了行结束符，因此如果不需要行结束符时还需要使用 rtrim()函数。

2. fgets()函数

fgets()函数从打开文件的指针中读取一行，语法格式如下所示：

```
string fgets(int $handle [, int $length])
```

在读取数据时遇到换行符（包括在返回值中）、EOF 或者已经读取了$length-1 字节后停止。如果没有指定$length，则默认为 1KB（1024 字节）。返回长度最多为$length-1 字节的字符串，如果出错则返回 false。

下面代码使用 fgets()函数读取 doc/game.txt 文件的内容并显示行号，运行效果与图 8-2 相同。

```
    <?php
        $filename="doc/game.txt";                    //指定文件
        $fp = fopen($filename, "r");                 //打开文件
        $line_num=0;                                 //指定初始行号
        do
        {
            echo "<li>NO #<b>{$line_num}</b> : " .fgets($fp) . "</li>";
                                                     //输出文件内容
            $line_num++;                             //行号递增
        }while (!feof($fp));                         //判断是否在文件最后
```

```
    fclose($fp);                                    //关闭文件
?>
```

由于 fgets()函数一次只能读取一行的内容，为了输出文件的所有内容，在上述代码的 do while 循环中使用 feof()函数判断是否还有内容。如果有则通过继续执行遍历所有行。

3. fgetss()函数

fgetss()函数功能与 fgets()函数基本相同，只是该函数在读取文件时会自动过滤 HTML 和 PHP 标记。语法格式如下：

```
string fgetss ( resource $handle [, int $length [, string $allowable_tags ]] )
```

下面的代码演示了如何使用 fgetss()函数读取 PHP 文件的内容：

```
<title>使用 fgetss()读取文件内容</title>
<h3>这一行会显示两次，第一次由 HTML 显示，第二次由 fegtss()函数读取显示</h3>
<?php
$filename = "fgetss.php";                    //指定文件
$fp = fopen ($filename, "r");                //打开文件
echo "这一行会执行，但不会由 fetss()函数输出，因为它在 PHP 代码中。<br/><br/>";
do {
    echo fgetss ($fp) . "<br/>";             //输出文件内容
} while (!feof($fp));                        //判断是否在文件最后
fclose ( $fp );                              //关闭文件
?>
```

保存上述代码为 fgetss.php，运行后的效果如图 8-3 所示。这是因为在 PHP 中的代码仅会在页面打开时执行，而在使用 fgetss()函数读取时会忽略其中的内容，对于 HTML 标记也忽略而只是输出 HTML 标记的内容。

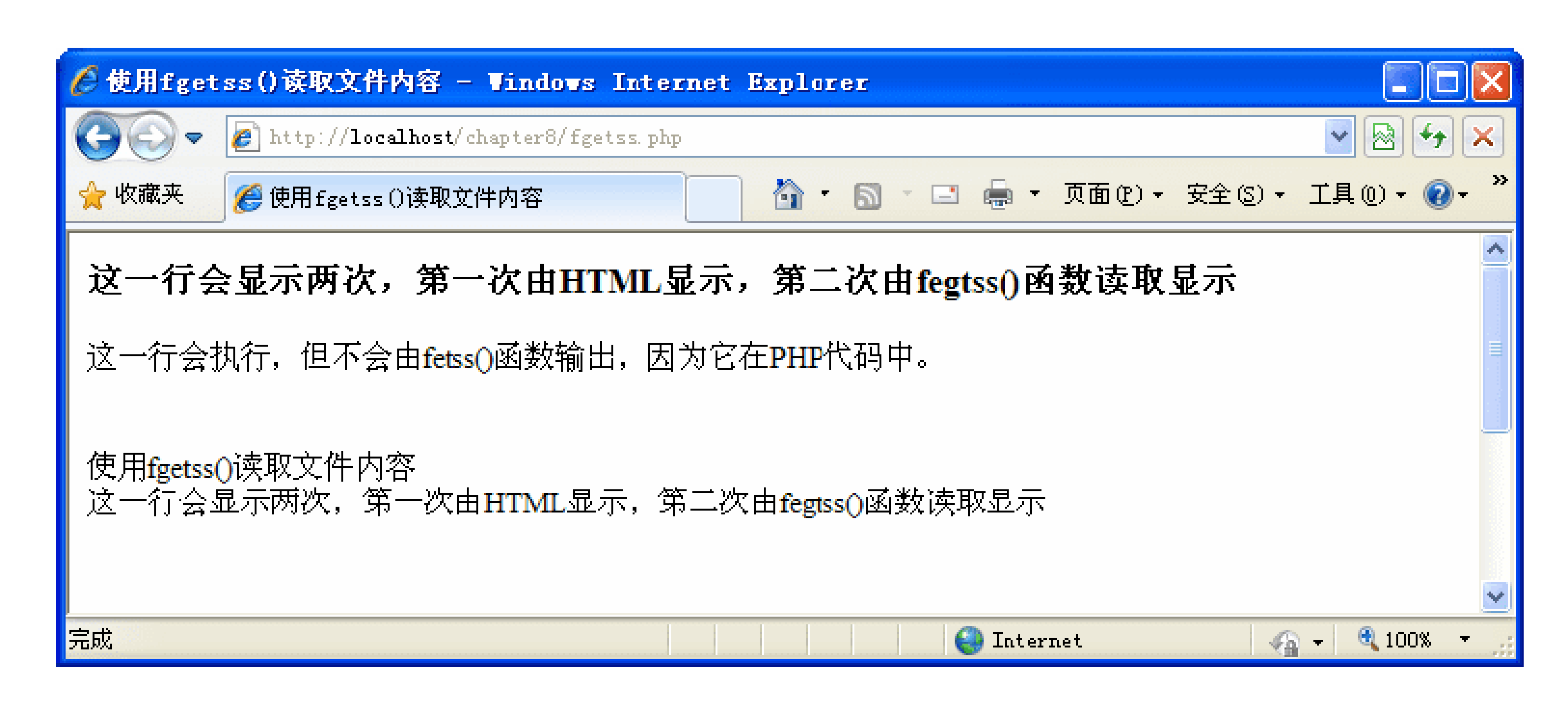


图 8-3　fgetss()读取文件内容效果

4. fgetcsv()函数

fgetcsv()函数用于从文件指针中读取一行，并且解析 CSV 字段，然后再返回一个包含这些字段的数组。fgetcsv()函数语法格式如下所示：

```
array fgetcsv ( int $handle [, int $length [, string $delimiter [, string $enclosure ]]] )
```

各个参数的说明如下所示。

- **$handle　一个有效的文件指针。**
- **$length　可选。用于指定行的最大长度，必须大于 CVS 文件内最长的一行。如果忽略该参数的话，那么长度就没有限制，不过可能会影响执行效率。**
- **$separator　可选。用于指定 CSV 的分隔符，默认值为逗号。**
- **$enclosure　可选。用于指定环绕符，默认值为双引号。**

fgetcsv()函数与 fgets()函数相似，不同的是 fgetcsv()解析读入的行并找出 CSV 格式的字段，然后返回一个包含这些字段的数组。在使用该函数时，如果出错，则返回 false。

【实践案例 8-3】

在网站 doc/data.txt 文件中保存了比赛的选手排名信息，包括名次、选手姓名、比赛场次和平均得分，其中每个信息之间使用逗号分隔，每行显示一条。例如，其中的内容如下所示：

```
排行,选手,场次,平均分
1,卢克-唐纳,52,9.81
2,泰格-伍兹,40,8.45
3,罗里-麦克,49,8.41
4,李-维斯特,46,7.89
5,韦伯-辛普,52,6.47
```

现在要求读取 data.txt 文件，并在页面上显示所有排名信息，具体步骤如下所示。

（1）首先在网站的 doc 目录中新建 trian.txt 文件，然后添加上述的内容。

（2）接下来在网站根目录中创建一个名为 fgetcsv.php 的 PHP 文件。

（3）在页面中使用 fgetcsv()函数读取 doc/data.txt 文件内容，实现代码如下：

```
<p class="h1">世界排名</p>
<table width="400" align="center">
<?php
$filename="doc/data.txt";                    //指定文件
$fp = fopen($filename,"r");                  //打开文件
$tag="th";
while ($ary = fgetcsv($fp, 1024, ","))      //逐行读取文件内容
{
    echo "<tr>";
    echo "<$tag>".$ary[0]."</$tag>";         //输出排名
    echo "<$tag>".$ary[1]."</$tag>";         //输出选手姓名
    echo "<$tag>".$ary[2]."</$tag>";         //输出比赛场次
    echo "<$tag>".$ary[3]."</$tag>";         //输出平均得分
    echo "</tr>";
    $tag="td";
}
```

```
    fclose($fp);                              //关闭文件
?>
    </table>
```

在浏览器中运行 fgetcsv.php 文件，在页面中会看到如图 8-4 所示的输出效果。

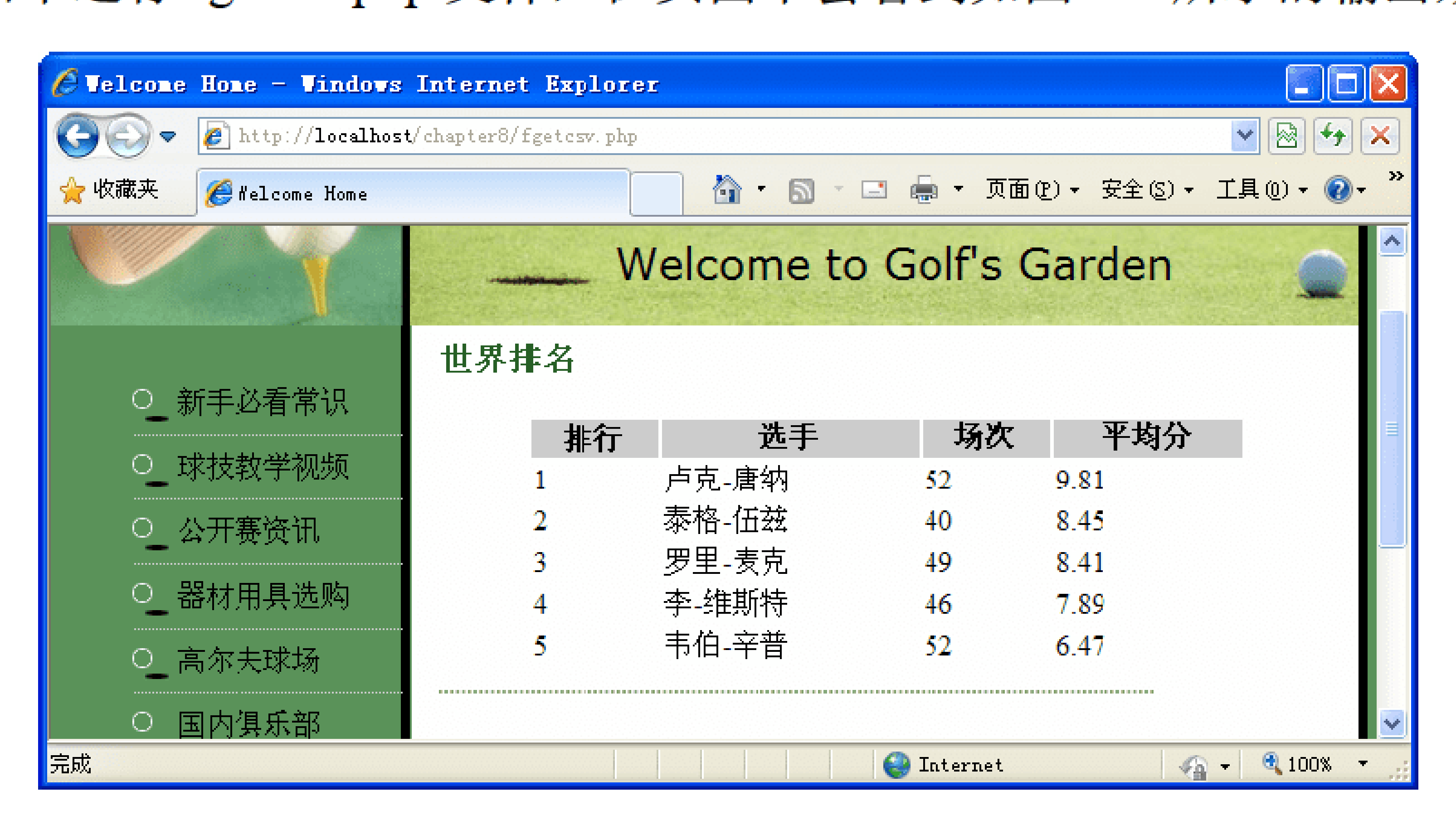


图 8-4　fgetcsv()读取文件内容效果

8.3.2　读取指定字节

使用按行读取函数可以方便地遍历整个文件，但是如果只需要读取其中一部分内容就需要使用按字节读取函数。PHP 提供了 3 个函数实现这个功能，分别是：fread()、fgetc()和 readfile()。

1. fread()函数

fread()函数可以读取已经打开的文件，并且可以指定要读取的字符数，语法格式如下所示：

```
string fread ( int $handle , int $length )
```

该函数从$handle 指定的资源中读取$length 个字符并返回，如果出错返回 false。当遇到下列情况之一时，读取将会终止。

- 当到达 EOF（文件末尾）时。
- 读取到$length 个字节时。
- 当一个包可用时。
- 已读取了 8192 个字节时。

例如，下面的代码演示了如何使用 fread()函数读取文件的全部内容和部分内容：

```
<?php
$filename = "doc/data.txt";                      //指定文件
$file = fopen($filename,"r");                    //打开文件
$getFile = fread($file,filesize($filename));     //读取全部内容
```

```
echo $getFile;                                              //输出
$filstext = fread($file,10);                                //读取 10 个字节到$filetext
echo $filstext;                                             //输出
fclose($file);                                              //关闭文件
?>
```


fread()函数与其他读取函数不同。使用该函数时，不用考虑换行符，因此只要使用 filesize()函数确定了文件的字节数，就能很方便地读取整个文件。

2. fgetc()函数

fgetc()函数可以从已经打开的文件指针中读取字符，并且只返回一个字符，语法格式如下所示：

```
string fgetc ( resource $handle )
```

返回$handle 在文件指针中的当前字符，如果遇到 EOF 则返回 false。这里需要注意，文件指针必须有效，并且必须指向一个由 fopen()或 fsockopen()成功打开的文件。

【实践案例 8-4】

假设有一个 doc/chars.txt 文件，里面的内容如下所示：

```
HELLO PHP
```

下面使用 fgetc()函数从 chars.txt 中读取一个字符，示例代码如下所示：

```
<?php
$filename=" doc/chars.txt";                                 //指定文件
$file = fopen($filename,"r");                               //打开文件
echo fgetc($file);                                          //读取一个字符，输出：H
fclose($file);                                              //关闭文件
?>
```

fgetc()函数同样可以遍历文件，下面代码实现从 chars.txt 文件中读取所有字符：

```
<?php
$filename="doc/chars.txt";                                  //指定文件
$file = fopen($filename,"r");                               //打开文件
do{
    echo fgetc($file);                                      //输出一个字符
}while(!feof($file));                                       //如果文件没有结束则继续
fclose($file);                                              //关闭文件
?>
```

3. readfile()函数

readfile()函数将读取一个文件并写入到输出缓冲。如果执行成功，则返回从文件中读取的字节数，失败则返回 false。语法格式如下所示：

```
int readfile ( string $filename [, bool $use include path [, resource
$context ]] )
```

例如，下面的代码演示了使用 readfile()函数读取 doc/chars.txt 文件的内容并输出：

```
<?php
$filename="doc/chars.txt";
$lenght = readfile($filename);          //读取 chars.txt 文件的内容到缓冲
echo "<br/>本次一共输出$lenght 字节。";
?>
```

8.3.3 读取全部内容

file_get_contents()函数可以一次性将整个文件的内容读入一个字符串中，语法格式如下所示：

```
string file_get_contents ( string $filename [, bool $use_include_path [,
resource $context [, int $offset [, int $maxlen ]]]] )
```

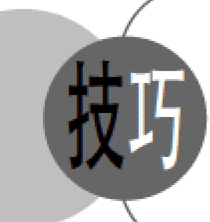
使用方法和 file()一样，不同的是 file_get_contents()函数将文件的内容读入到一个字符串中。函数的$offset 参数表示读取文件时开始读取的位置，$maxlength 表示此次需要读取的字节数。

【实践案例 8-5】

在网站的 doc 目录有一个 module.html 文件，其中保存的是外部模块的 HTML 代码。下面使用 file_get_contents()函数读取该文件并将 HTML 代码显示到页面。实现代码如下所示：

```
<p class="h1">查看外部模块的 HTML 代码</p>
<p>
<?php
    $filepath="doc/module.html";        //指定文件路径
    $contents = file_get_contents($filepath);
                                        //读取文件内容，保存到$contents 中
    echo "doc/module.html 文件内容如下：<br/><br/>";
    echo htmlspecialchars($contents);   //输出读取的文件内容
?>
</p>
```

file_get_contents()函数的作用是将文件读取到一个字符串，因此整个文件的内容将作为这一个字符串来处理。为不使字符串中的 HTML 标记被解析，使用 htmlspecialchars()函数进行处理，运行后的效果如图 8-5 所示。



file_get_contents()函数是用来将文件的内容读入到一个字符串中的首选方法。如果操作系统支持还会使用内存映射技术来增强性能。

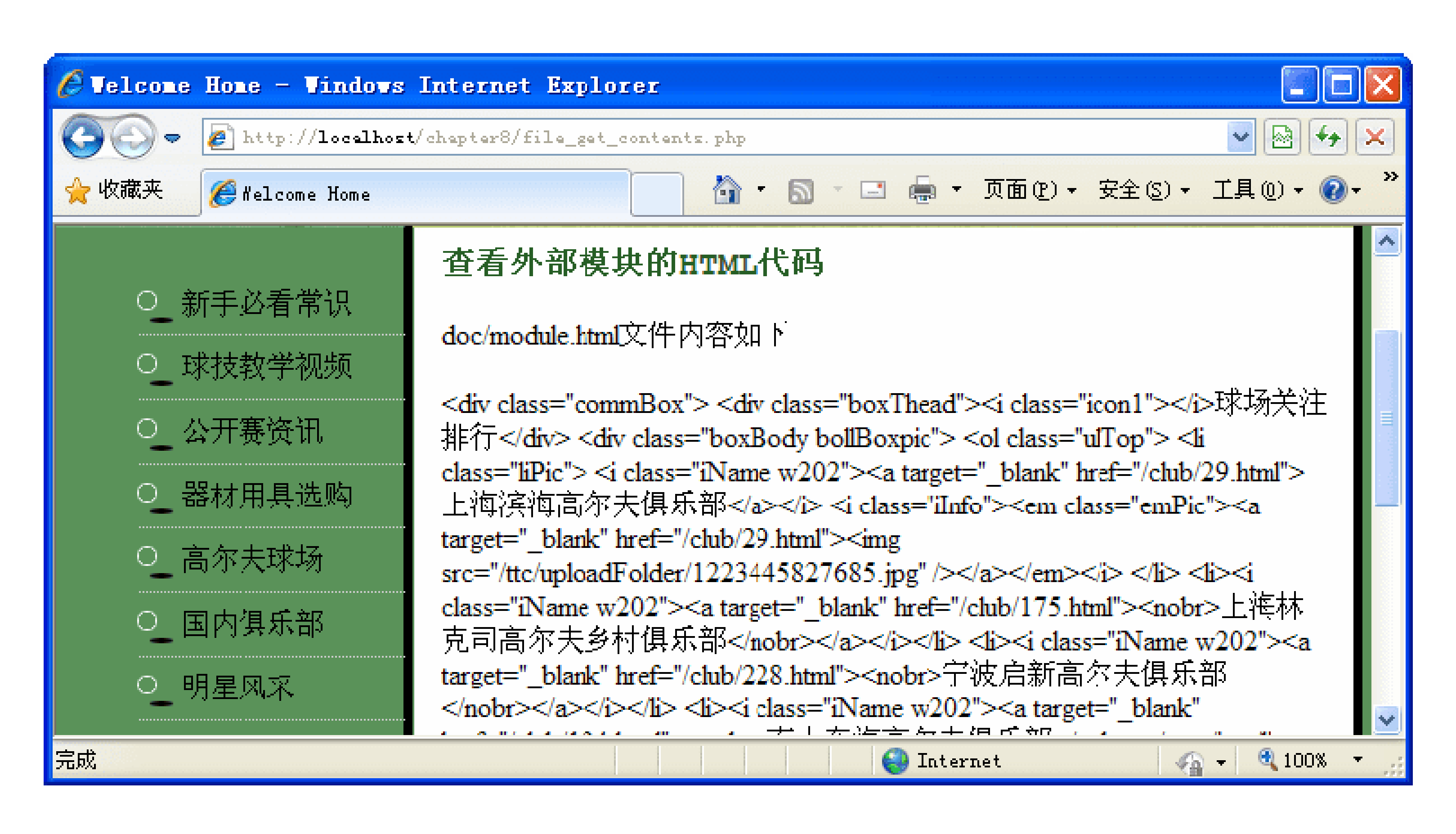


图 8-5　使用 file_get_contents()函数读取文件内容

8.3.4　其他读取函数

除了前面介绍的文件读取函数之外，在 PHP 中还经常使用 fpassthru()函数读取文件内容。fpassthru()函数输出文件指针处的所有剩余数据，语法格式如下所示：

```
int fpassthru ( resource $handle )
```

该函数将给定的文件指针从当前的位置读取到 EOF 并把结果写到输出缓冲区。如果发生错误返回 false，否则返回从$handle 读取并传递到输出的字符数。

下面的代码演示了如何使用 fpassthru()函数读取 doc/chars.txt 文件的内容：

```
<p class="h1">使用 fpassthru()函数</p>
<p>
<?php
$filename="doc/chars.txt";                    //指定文件路径
$file = fopen($filename,"r");                 //打开文件
echo "第一行的内容: ".fgets($file);           //读取第一行
echo "<hr/>下面是调用 fpassthru()函数返回的内容: <br/>";
$length=fpassthru($file);                     //把文件的其余部分发送到输出缓存
echo "<br/>调用 fpassthru()函数一共输出了 $length 字符。";
fclose($file);                                //关闭文件
?>
</p>
```

在上述代码中，首先使用 fgets()函数从 chars.txt 中读取了一行，然后使用 fpassthru()函数把文件的其余部分发送到输出缓存。利用 fpassthru()函数的返回值，还显示了输出的字符数。

【实践案例 8-6】

fpassthru()函数还有一个特殊的用法，就是读取二进制文件。例如可以显示一张图片或者下载 XLS 文件等。下面代码演示了以二进制格式读取一张图片并显示到页面的代码：

```
<?php
$filename = "images/ad.gif";                    //指定图片路径
$fp = fopen($filename, 'rb');                   //以二进制打开方式
header("Content-Type: image/gif");              //设置图片的类型
header("Content-Length: " . filesize($filename));      //设置图片的大小
fpassthru($fp);                                 //将图片的二进制流输出到页面
exit;                                           //终止程序
?>
```

在这里要注意，读取二进制文件时应该为 fopen()函数的第二个参数附加 b 标记。将上述代码保存为 fpassthru.php，运行后的效果如图 8-6 所示。

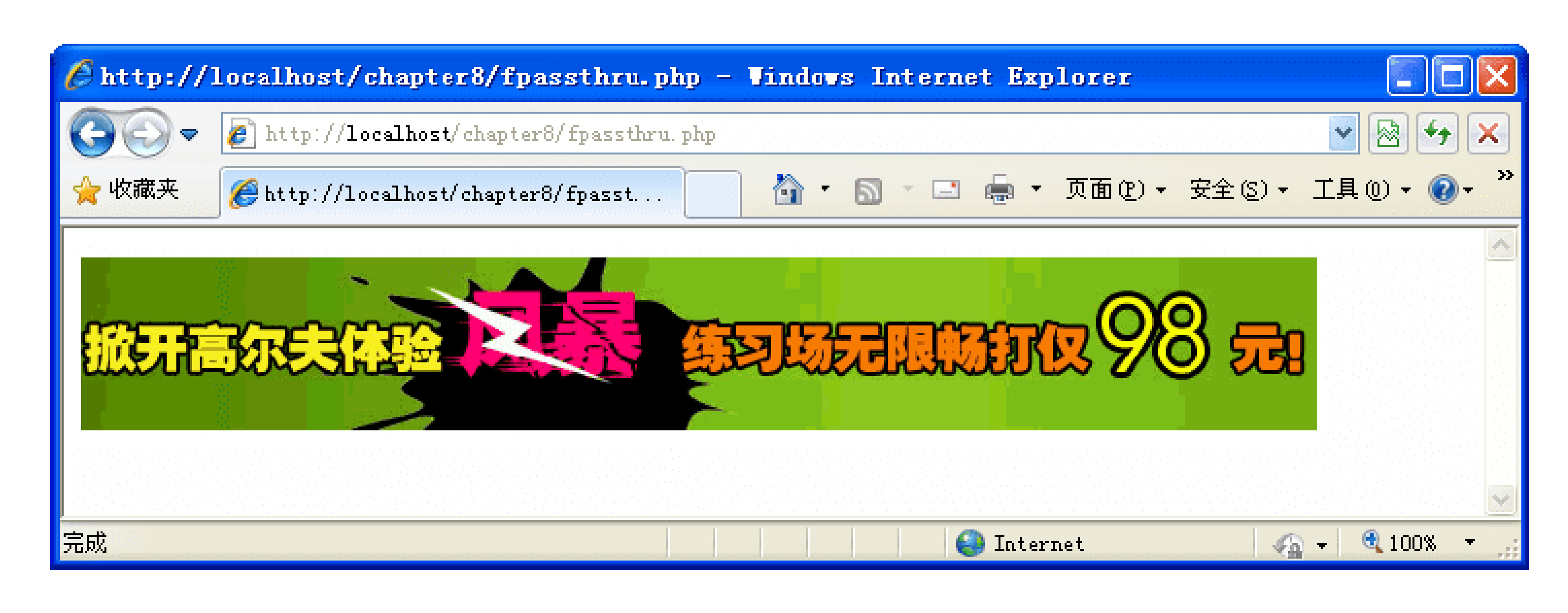


图 8-6　读取图片文件

8.4　移动文件指针

在读取文件时经常需要在文件中进行跳跃式的访问，例如要输出奇数行的内容，而非逐行读取，这时候就需要在读取第一行内容之后将文件指针移动到第三行。为了实现上述功能，PHP 提供了 3 个移动文件指针的函数，本节主要讲解它们。

8.4.1　fseek()函数

fseek()函数用于在打开的文件中把文件指针从当前位置向前或向后移动到新的位置。新位置从文件头开始以字节数为单位，成功则返回 0，否则返回-1。语法格式如下所示：

```
int fseek ( resource $handle , int $offset [, int $whence ] )
```

该函数将$handle 指针移动到$offset 指定的位置，如果忽略可选参数$whence，则位置将设置为从文件开始到 offset 字节处。$whence 将影响指针的位置，有如下 3 个可选的值。

- **SEEK_CUR　设置指针位置为当前位置加上$offset 字节。**
- **SEEK_END　设置指针为 EOF 加上$offset 字节，$offset 必须设置为负值。**
- **SEEK_SET　设置指针为$offset 字节处，这与忽略$whence 效果相同。**

例如，下面的代码演示了如何使用 fseek()函数：

```
<?php
$filename="doc/chars.txt";
$file = fopen($filename,"r");        //打开文件
echo fgets($file);                   //读取第一行
fseek($file,0);                      //将文件指针移到文件开头
echo fgets($file);                   //再次读取第一行
fclose($file);                       //关闭文件
?>
```

在上述代码中，首先调用 fgets()函数读取文件第一行的内容，此时指针在第二行的开始。然后，调用 fseek()函数将文件指针移到文件的开头，即第一行的开始处。因此，当再次调用 fgets()函数读取文件一行内容的时候，实际上是输出第一行的内容。

下面代码演示了 fseek()函数的其他用法：

```
fseek($file,100,SEEK_CUR);      //将文件指针从当前位置向前移动 100 字节
fseek($file,-100,SEEK_CUR);     //将文件指针从当前位置向后移动 100 字节
fseek($file,100,SEEK_END);      //将文件指针从当前位置移动到末尾的前 100 字节处
```

8.4.2 ftell()函数

ftell()函数主要用于获取打开文件指针的当前位置，语法格式如下所示：

```
int ftell(resource $handle)
```

执行后返回由$handle 指定的文件指针的位置，也就是文件流中的偏移量。如果出错则返回 false。

例如，下面的代码演示了如何使用 ftell()函数：

```
<?php
$filename="doc/chars.txt";
$file = fopen($filename,"r");                   //打开文件
echo "文件指针当前位置: ".ftell($file);          //输出此时文件的位置，为 0
fseek($file,150);                               //移动文件指针
echo "<br/>";
echo "文件指针当前位置: ".ftell($file);          //输出此时文件的位置，为 150
fclose($file);                                  //关闭文件
?>
```

执行结果如下所示：

```
文件指针当前位置: 0
文件指针当前位置: 150
```

8.4.3 rewind()函数

rewind()函数将文件指针定位到打开文件的开头，语法格式如下所示：

```
bool rewind(resource $handle)
```

如果执行成功则返回 true，否则返回 false。下面的代码演示了如何使用 rewind()函数：

```
<?php
$filename="doc/chars.txt";
$file = fopen($filename,"r");                //打开文件
fseek($file,15);                             //移动文件指针
if(rewind($file)) {                          //将指针移到文件开始
    echo "文件指针已经回到了文件的开头";
}
else {
    echo "文件执行失败";
}
fclose($file);                               //关闭文件
?>
```

8.5 写入文件

读取文件不会改变文件的内容，如果要实现修改文件的功能，必须对文件进行写入操作。在 PHP 中可通过 fwrite()函数、fputs()函数 file_put_contents()函数和来完成文件的写入。

8.5.1 fwrite()函数

fwrite()函数主要用于写入文件，语法格式如下所示：

```
int fwrite(resource $handle, string $string [, int $length])
```

fwrite()把$string 的内容写入文件指针$handle 处。如果指定了$length，当写入了$length 字节或者写完了$string 以后，写入就会停止。该函数返回写入内容的字节数，如果出错则返回 false。

【实践案例 8-7】

假设在网站 doc 目录下有一个 zhuanti.txt 文件，编写程序向该文件中写入内容并读取到页面显示。

（1）首先在 PHP 中定义一个字符串，在其中保存要写入的内容。

```
<?php
$str=<<<golf
[赛事专题]
2012 沃尔沃中国公开赛(12 月 15 日 10:53)
2012 美国高尔夫公开赛(12 月 15 日 10:53)
2012 美国大师赛(12 月 15 日 10:53)
2012 英国公开赛(12 月 15 日 10:53)
```

```
2011 中国高尔夫球业余公开赛 (10 月 18 日 16:05)
WGC 世界业余高尔夫锦标赛(10 月 18 日 16:05)
2010 年英国公开赛(06 月 29 日 11:59)
golf;
?>
```


（2）使用 fwrite()函数将$str 指定的内容写入 doc/zhuanti.txt 文件，具体实现代码如下：

```
<?php
$filename = 'doc/zhuanti.txt';                  //指定文件路径
if (is writable($filename)) {
    if (!$handle = fopen($filename, 'a')) {//判断文件是否能正常打开
        echo "不能打开文件 $filename";
        exit;
    }
    $sizes=fwrite($handle, $str);               //将$str 写入到打开的文件中
    if ($sizes=== false) {
        echo "不能写入到文件 $filename";
        exit;
    }
    echo "成功写入到文件$filename ，本次写入$sizes 字节数。写入内容如下：";
    fclose($handle);                            //关闭文件
} else {
    echo "文件 $filename 不可写";
}?>
```

如上述代码所示，首先使用 is_writable()函数确定具有文件的写入权限，然后使用 fopen()函数打开文件将文件指针保存到$handle。文件打开之后指针默认位于文件开头，这也是要写入的位置。接下来用 fwrite()函数将$str 写入到当前指针位置（即文件开头），并将写入字节数保存到$sizes 变量。最后，输出信息并关闭文件。

（3）如果没有错误上面代码已经实现向文件中写入内容。要读取文件内容有很多函数，这里使用 fgets()实现，代码如下所示：

```
   <p class="h1">赛事进程</p>
    <ul class="t1">
<?php
$fp = fopen($filename, "r");                    //打开文件
do{
    echo "<li> " .fgets($fp) . "</li>";         //输出文件内容
}while (!feof($fp));                            //判断是否在文件最后
fclose($fp);                                    //关闭文件
?>
</ul>
```

（4）将文件保存为 fwrite.php，在浏览器中运行将看到页面中的输出信息，如图 8-7 所示。



图 8-7　写入文件运行效果

使用 fwrite()函数写入时要注意不同类型的操作系统具有不同的行结束符号。因此，在写入内容中想插入一个新行时，就需要使用符合操作系统的行结束符号。基于 UNIX 的系统使用\n 作为行结束字符，基于 Windows 的系统使用\r\n 作为行结束字符，基于 Macintosh 的系统使用\r 作为行结束字符。

8.5.2　fputs()函数

fputs()函数的功能和 fwrite()函数一样，并且用法也相同，fputs()函数的语法格式如下所示：

```
int fputs(resource $handle, string $string [, int $length])
```

【实践案例 8-8】

下面使用 fputs()函数向 doc/zhuanti.txt 文件末尾追加内容，并输出写入的字节和写入后的文件内容。

如下所示是写入内容的代码：

```
 <p class="h1">赛事进程</p>
<?php
$str=<<<golf                                //要写入的字符串
\r\n[产业专题]
球童？球童！(07 月 12 日 17:04)
高尔夫建设，在艰难中探索(07 月 01 日 13:23)
高尔夫成功入奥(05 月 12 日 16:21)
golf;

$filename = 'doc/zhuanti.txt';              //指定文件名
if (is_writable($filename)) {               //判断是否可写
    $handle = fopen($filename, 'a');        //打开文件
```

```
        fseek($handle,0,SEEK_END);            //将文件指针移到文件最后
        $sizes=fputs($handle, $str);          //写入内容
        echo "成功写入到文件$filename ，本次写入$sizes 字节数。写入内容如下：";
        fclose($handle);                      //关闭文件
    } else {
        echo "文件 $filename 不可写";
    }   ?>
```

上面仅演示了 fputs()函数的写入代码，读取显示的代码与前面相同，这里就不再重复。需要注意的是，由于这里是向文件末尾写入内容，所以文件打开之后必须移动文件指针。将文件保存为 fputs.php，运行将看到如图 8-8 所示的效果。

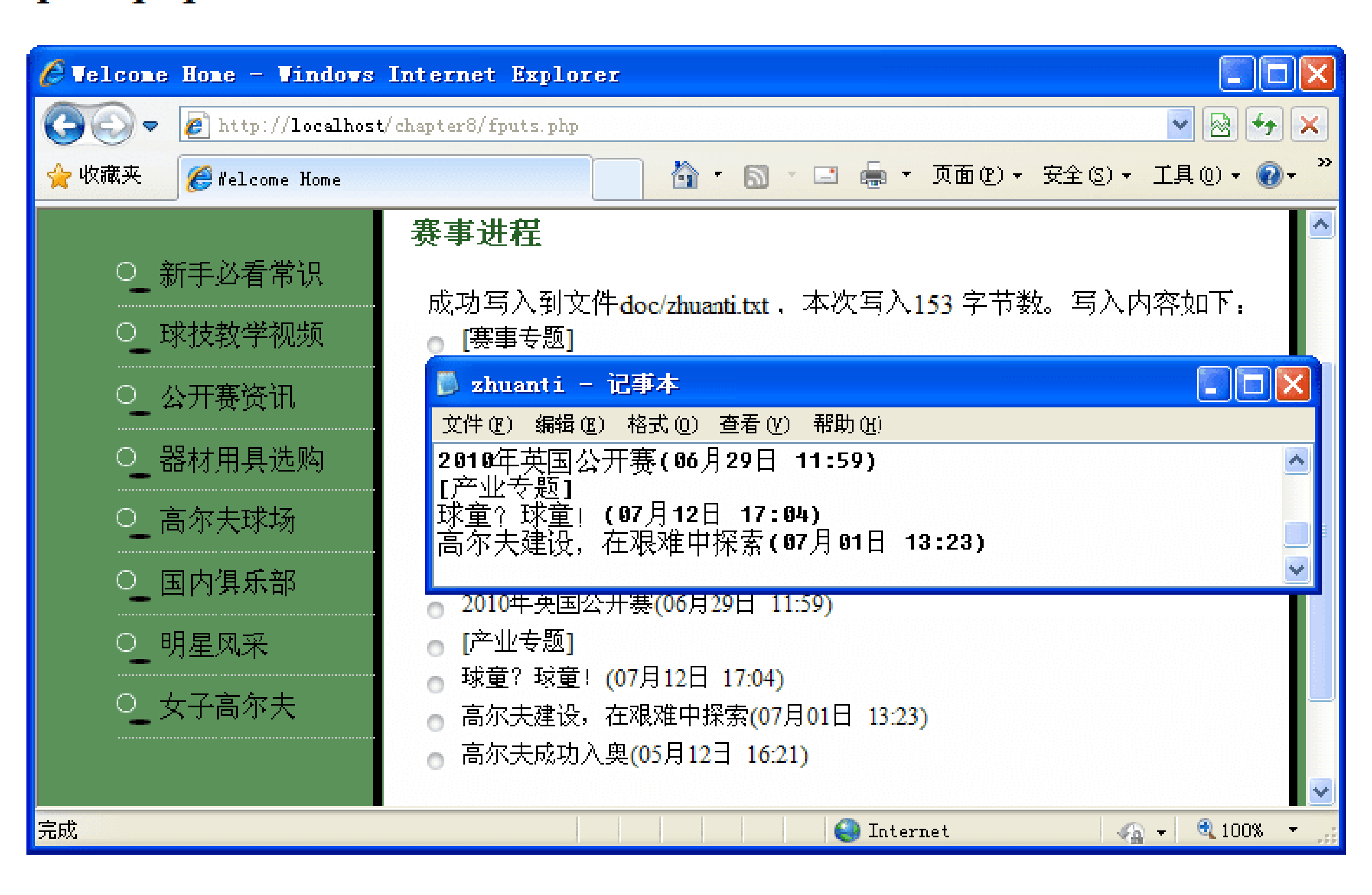


图 8-8　fputs()函数写入文件效果

8.5.3　file_put_contents()函数

在使用 fwrite()和 fputs()写入时，必须先调用 fopen()函数打开文件，完成之后还要调用 fclose()函数关闭文件。而 file_put_contents()函数可以使这个过程自动化，只需指定要写入的字符串即可。语法格式如下：

```
int file_put_contents ( string $filename , string $data [, int $flags [,
resource $context ]] )
```

其中，$filename 参数是要写入的文件名，$data 参数是写入内容（也可以是数组，但不能为多维数组），$flags 是写入时的标识，可以为 FILE_USE_INCLUDE、FILE_APPEND 或者 LOCK_EX，$context 是一个 context 资源。

file_put_contents()函数执行成功后将返回写入到文件内的字节数，失败时返回 false。

例如，下面的示例代码演示了如何使用 file_put_contents()函数向文件内写入字符串和数组：

```
<p class="h1">创建临时文件</p>
```

```
<?php
$str=<<<golf                                                //定义字符串
[球具装备]
最新最权威的评测和选购技巧，海量数据助你成功\r\n
golf;
$tools=array(0=>"Kia Ma Doytona 推杆",1=>"PING I20 发球木",2=>"TaylorMade R11S
发球木",3=>"Odyssey Protype TOUR Series 9 推杆");           //定义数组
    $filename = 'doc/temp.txt';
    $sizes=file_put_contents($filename,$str);               //写入字符串
    echo "本次写入了 $sizes 字节";
    file_put_contents($filename,$tools,FILE_APPEND);        //以追加方式写入数组
?>
```

将上述代码保存为 file_put_contents.php，运行效果如图 8-9 所示。

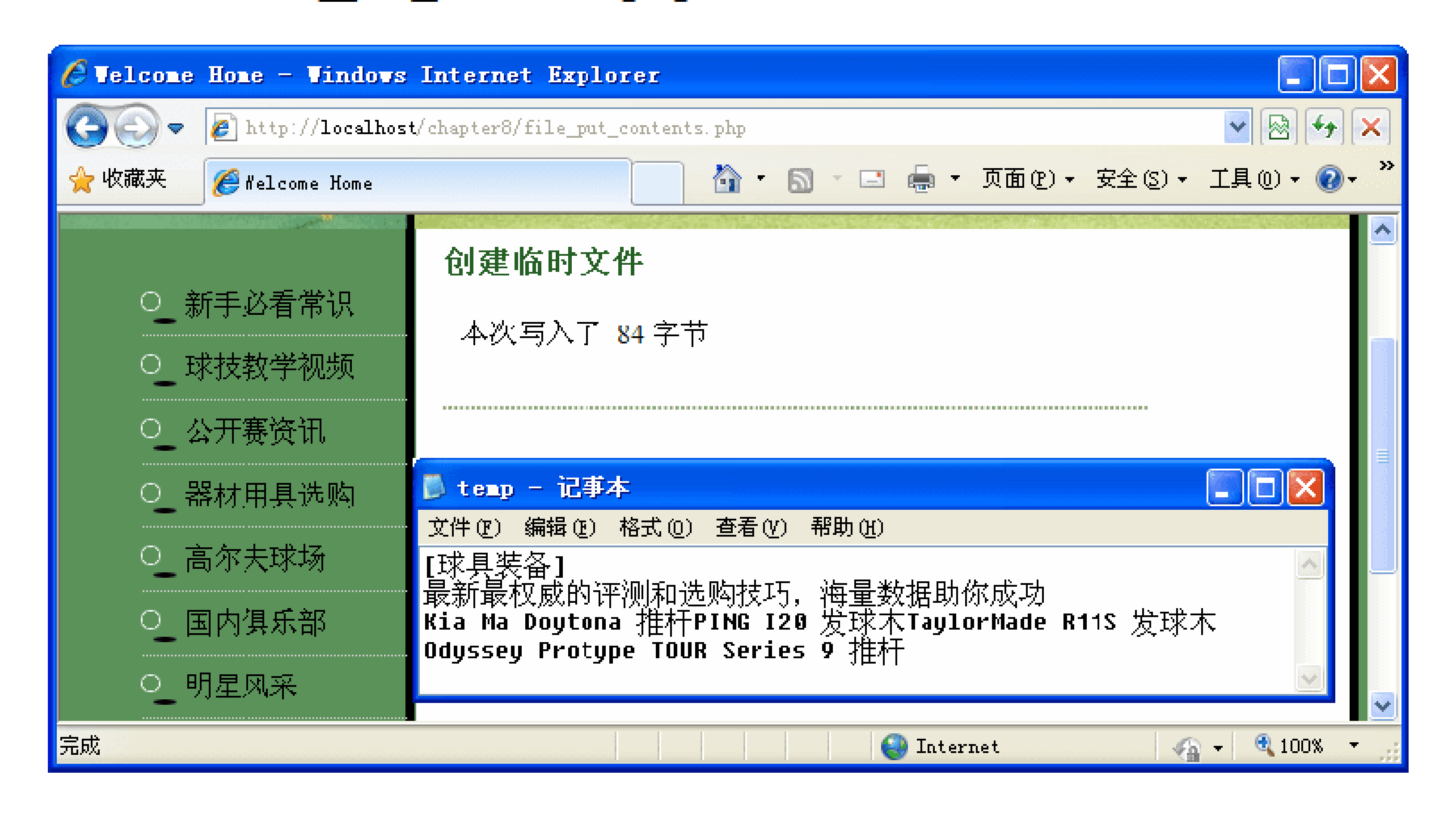


图 8-9　file_put_contents ()函数写入文件效果

8.6　操作文件

本节主要介绍如何使用 PHP 系统函数实现对文件的复制、重命名和删除操作。

8.6.1　复制文件

PHP 的 copy()函数实现了复制文件功能，它的语法格式如下所示：

```
bool copy ( string $source , string $dest )
```

执行后它将文件从$source 复制到$dest，成功返回 true，否则返回 false。

例如，下面的代码使用 copy()函数将 doc/chars.txt 的内容复制到 doc/string.txt 文件：

```
<?php
$filename="doc/chars.txt";
```

```
$newfilename="doc/string.txt";
if(copy($filename,$newfilename)) {
    echo "复制成功";
}else {
    echo "复制失败";
}
?>
```



如果复制一个零字节的文件，copy()将返回 false，但文件仍然会被正确复制。

8.6.2 重命名文件

要对一个文件进行重命名操作可以使用 rename()函数，语法格式如下：

```
bool rename ( string $oldname , string $newname [, resource $context ] )
```

执行时该函数会尝试将$oldname 文件重命名为$newname 指定的名称，成功时返回 true，否则返回 false。

例如，下面的代码使用 rename()函数将 doc/chars.txt 文件重命名为 doc/all.txt 文件：

```
<?php
$oldname="doc/chars.txt";
$newname="doc/all.txt";
if(rename($oldname,$newname)){
    echo "chars.txt 已经更改为 all.txt";
}else {
    echo "对 chars.txt 的重命名失败";
}
?>
```

8.6.3 删除文件

要删除一个文件可以使用 unlink()函数，它的语法格式如下所示：

```
bool unlink ( string $filename )
```

执行后将它删除$filename 指定的文件，成功返回 true，否则返回 false。

例如，下面的代码演示了如何使用 unlink()函数删除文件：

```
<?php
$filename="doc/chars.txt";                  //指定文件
if (file_exists($filename)) {               //判断文件是否存在
```

```
    if(unlink($filename)) {          //执行删除操作
        echo "文件删除成功";          //删除成功
    }
    else {
        echo "文件删除失败";          //删除失败
    }
}
?>
```

8.7 操作目录

之前介绍的都是使用 PHP 系统函数对文件的操作，像获取文件属性、打开/关闭文件、读取/写入文件等。PHP 同样提供了使用系统函数操作目录的功能，可以实现对目录的遍历、对目录的修改和对目录的删除等操作。

8.7.1 打开目录

在 PHP 中 opendir()函数用于打开一个目录，可用于之后的 closedir()函数和 readdir()函数中。该函数的语法格式如下所示：

```
resource opendir(string path [,resource $context])
```

执行该函数，如果执行成功返回一个目录句柄，否则返回 false。在 opendir()前面加上“@”符号来隐藏错误信息的输出。

例如，下面的代码使用 opendir()函数尝试打开 C:/windows 目录：

```
<?php
$path="C:/windows";
if(is_dir($path))
{
    if(opendir($path)){
        echo "打开  $path 目录成功";
    }else{
        echo "打开 $path 目录失败";
    }
}
else{
    echo $path."不是一个目录";
}
?>
```

8.7.2 关闭目录

打开的目录使用完毕之后就需要关闭，在 PHP 中使用 closedir()函数关闭打开的目录。

该函数的语法格式如下所示：

```
void closedir(resource $dir_handle)
```

closedir()函数关闭由$dir_handle 指定的目录句柄，但是句柄必须是由 opendir()函数打开的。

例如，下面使用 opendir()函数和 closedir()函数创建一个示例，演示如何打开和关闭目录，具体代码如下所示：

```
<?php
$dirname="files/";                    //指定一个目录名称
if(is_dir($dir))                      //判断是否为目录
{
    $d= opendir($dir);                //打开目录
    closedir($d);                     //关闭目录
}
?>
```

8.7.3　遍历目录

打开目录之后便可以对它进行操作，最常见的就是遍历目录的内容。在 PHP 中可以使用 readdir()和 scandir()两个函数实现这个功能。

1. readdir()函数

readdir()函数返回由 opendir()打开目录句柄中的内容，函数语法格式如下：

```
string readdir(resource $dir_handle)
```

如果该函数执行成功，则返回一个文件名，根据这个文件名可以列出目录中的所有文件；否则返回 false。

【实践案例 8-9】

假设在网站下有一个 Files 目录，它是用户上传的根目录，现在编写代码读取该目录中的内容，并显示文件名称、类型和大小。实现代码如下所示：

```
<p class="h1">查看上传目录内容</p>
<table width="400" align="center">
    <tr>
    <th>文件名称</th>
    <th>类型</th>
    <th>大小</th>
    </tr>
<?php
$dirname = "Files/";                              //指定目录位置
if (is_dir($dirname))                             //判断目录是否存在
{
```

```
    if ($dh = opendir($dirname))   {                //打开目录
        while (($file = readdir($dh)) !== false) {         //读取文件列表
            echo "<tr><td> $file </td>";
            if(filetype($dirname . $file)=="dir"){       //判断类型
                echo "<td>目录</td><td>0</td>";
            }
            else{
                echo "<td>文件</td>";
                echo "<td>".filesize($dirname . $file)."</td>";
            }
            echo "</tr>";
        }
        closedir($dh);                           //关闭目录
    }
}
?>
</table>
```

保存上述代码为 readdir.php，运行后的效果如图 8-10 所示。

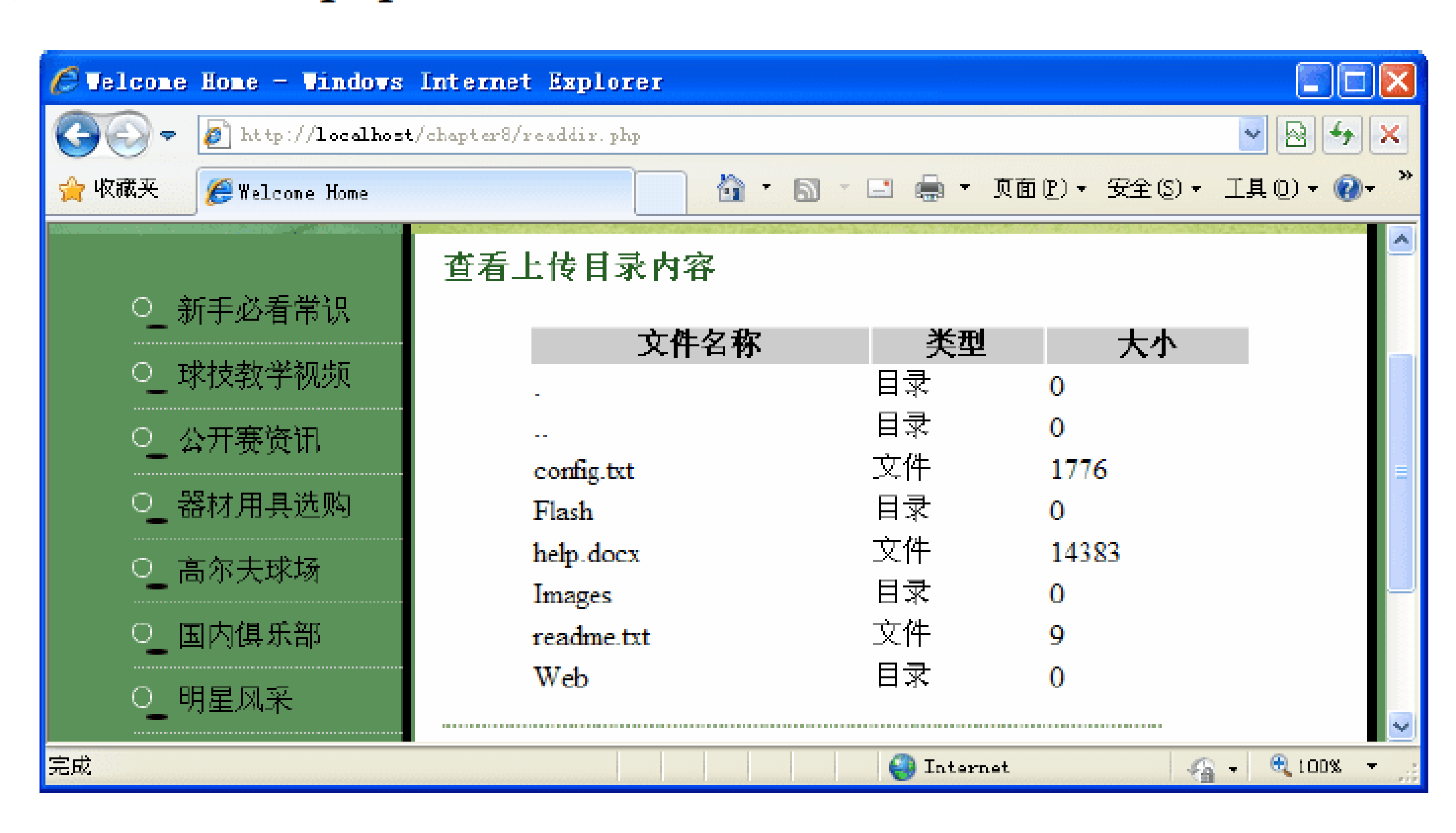


图 8-10　readdir()函数读取目录内容效果

2. scandir()函数

scandir()函数执行后会返回包含文件和目录的数组，否则返回 false。语法格式如下所示：

```
array scandir ( string $directory [, int $sorting order [, resource
$context ]] )
```

其中，$directory 参数表示要遍历的目录；$sorting_order 参数用于指定默认的排序顺序是按字母升序排列，如果设为 1 则按字母降序排列；$context 参数用于设置如何配置目录句柄。

【实践案例 8-10】

假设在这里要使用 scandir()函数来读取 Files 目录的内容，实现代码如下所示：

```
<h2>使用 scandir()函数读取目录内容</h2>
<?php
$dirname = "Files/";                //指定目录
$files = scandir($dirname);         //调用 scandir()函数，将结果保存到$files 数组
echo "<pre>";
print r($files);                    //输出数组的内容
echo "</pre>";
?>
```

保存上述代码为 scandir.php，运行后的效果如图 8-11 所示。

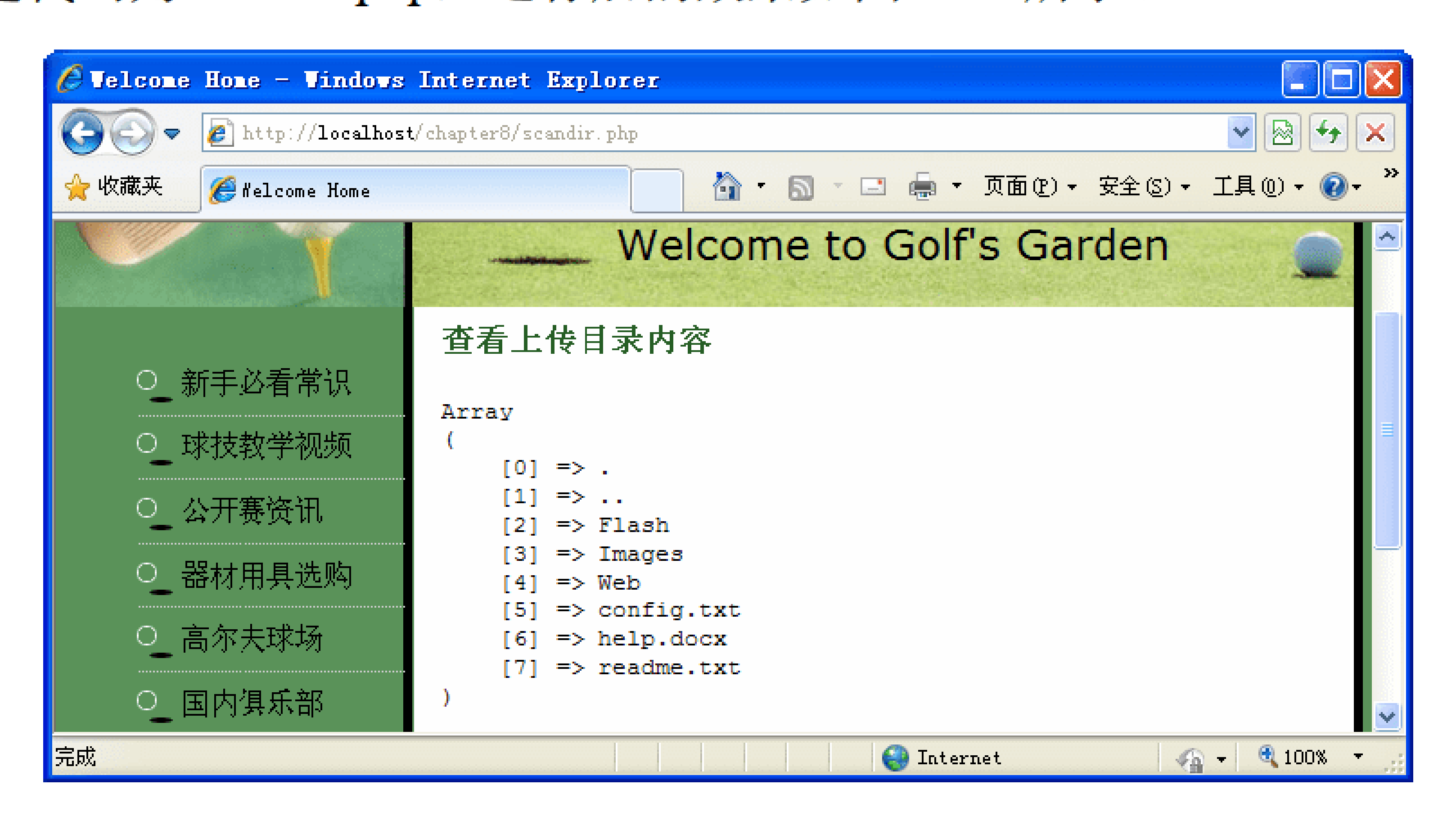


图 8-11　scandir()函数读取目录内容效果



对于非目录的路径调用 scandir()函数将返回 false，并生成一条 E_WARNING 级的错误。

8.7.4　创建目录

要创建一个目录可以调用 mkdir()函数，该函数语法格式如下：

```
bool mkdir ( string $pathname [, int $mode [, bool $recursive [, resource
$context ]]] )
```

执行后将尝试新建由$pathname 参数指定的目录，且默认的$mode 是 0777，即具有最大的访问权限。新建成功时返回 true，否则返回 false。

例如，下面的代码演示了如何使用 mkdir()函数新建目录：

```
<?php
$newdirname="doc/newdir";               //指定新目录路径
if(!mkdir($newdirname)){                //创建目录
    echo "目录创建失败";
}else{
    echo $newdirname."目录创建成功";
```

```
}
?>
```

在创建新目录时还可以指定访问权限，下面代码为新建的 db 目录指定 0700 权限：

```
mkdir("mydoc/db",0700);
```

8.7.5 删除目录

rmdir()函数的作用与 mkdir()函数相反，rmdir()函数可以删除一个目录。rmdir()函数语法格式如下：

```
bool rmdir ( string $dirname )
```

执行时将尝试删除$dirname 所指定的目录。如果该目录为空，且有相应的权限，则删除成功返回 true，否则返回 false。

下面代码使用 rmdir()函数删除 doc/newdir 目录：

```
<?php
$dirname=" doc/newdir ";                       //指定目录
if (is_dir($filename)) {                       //判断目录是否存在
    if(rmdir($dirname)) {                      //执行删除操作
        echo "目录删除成功";                   //删除成功
    }else {
        echo "目录删除失败";                   //删除失败
    }
}
?>
```

8.8 解析路径

PHP 提供了一些函数可以实现对文件路径的操作。例如，获取路径中的目录，或者路径中文件名称以及扩展名。

8.8.1 获取文件名

basename()函数用于获取路径中的文件名部分，语法格式如下所示：

```
string basename(string $path [,string $suffix])
```

这里的$path 参数表示需要解析的路径；$suffix 为可选参数，表示文件扩展名。如果提供了$suffix 参数，不会输出这个扩展名。

下面的代码从 files/images/logo.gif 路径中分别获取带扩展名和不带扩展名的文件名称。

```
<?php
$path = "files/images/logo.gif";
$file = basename($path);                    //$file 的值为 logo.gif
$file = basename($path,".gif");             //$file 的值为 logo
?>
```


8.8.2　获取目录部分

dirname()函数的作用与 basename()函数类似，dirname()函数可以从路径中获取目录部分。dirname()函数语法格式如下所示：

```
string dirname(string $path)
```

其中的$path 参数是指向一个文件的全路径字符串，返回去掉文件名后的目录名。下面的代码演示了如何使用 dirname()函数获取路径中的目录名称：

```
<?php
$path ="files/images/logo.gif ";
echo dirname($path);                            //输出 files/images
echo dirname('C:\Windows\system32\boot.ini');   //输出 C:\Windows\system32
echo dirname('c:/boot.ini');                    //输出 c:\
?>
```

8.8.3　获取路径中的各个部分

basename()函数仅适用于返回文件名称，而 dirname()函数仅适用于返回目录部分。如果要返回路径中的所有部分应该怎么办呢？

PHP 的 pathinfo()函数会以数组的形式返回路径信息，语法格式如下所示：

```
mixed pathinfo ( string $path [, int $options ] )
```

其中，$path 参数表示路径字符串，返回的关联数组中包括 path 的 4 个部分：目录名、基本名、扩展名和文件名，分别通过 dirname、basename、extension 和 filename 索引来引用。通过可选参数$options 可以指定要返回哪些部分。

【实践案例 8-11】

假设，要获取“files/images/logo.gif”路径中的各个部分。使用 pathinfo()函数的实现代码如下所示：

```
<?php
$path =" files/images/logo.gif ";
$pathinfo = pathinfo($path);
print_r($pathinfo);
echo "拆分数组后的输出: \n";
echo "目录名: ".$pathinfo['dirname']."\n";
echo "基本名: ".$pathinfo['basename']."\n";
```

```
echo "扩展名: ".$pathinfo['extension']."\n";
echo "文件名: ".$pathinfo['filename'];
?>
```

执行后将会输出如下结果：

```
Array
(
    [dirname] =>  files/images
    [basename] => logo.gif
    [extension] => gif
    [filename] => logo
)
拆分数组后的输出:
目录名:  files/images
基本名: logo.gif
扩展名: gif
文件名: logo
```

8.8.4 获取绝对路径

realpath()函数可以将网站目录下的某一个相对路径转换成绝对路径，语法格式如下所示：

```
string realpath(string $path)
```

执行时会将$path 参数中的所有路径和目录以及相对路径引用转换为相应绝对路径。如果执行失败则返回 false，例如文件不存在时。

例如，要获取 index.php 文件所在的绝对路径，可以使用如下的代码：

```
<?php
$path = "index.php";                 //指定要获取绝对路径的相对路径引用
echo realpath($path);                //输出结果
?>
```

执行后的输出结果如下：

```
D:\PHPSpace\Chapter8\index.php
```

realpath()函数中支持相对路径的写法，下面的方法都是正确的：

```
realpath("doc/web/index.php");
realpath("/doc/web/index.php");
realpath("../images/logo.gif ");
realpath("../index.php");
```

8.9　读取磁盘属性

除了前面介绍的针对文件和目录的操作外，PHP 还提供了两个函数用于读取指定目录所在磁盘的可用空间和总空间大小。

8.9.1　获取目录所在磁盘的可用空间

disk_free_space()函数可以以字节为单位返回指定目录所在磁盘分区的可用空间。该函数语法格式如下所示：

```
float disk_free_space(string $diectory)
```

例如，要获取网站根目录所在磁盘和系统盘的可用空间，可用如下代码：

```
<?php
$drive="/";                                   //指定网站根目录
$freespace=disk_free_space($drive);     //获取可用空间
echo "网站所在磁盘的剩余空间为:".formatFreeSpace($freespace);
$cfree=disk_free_space("C:");               //获取 C 盘可用空间
echo "\n 系统盘的剩余空间为: ".formatFreeSpace($cfree);

function formatFreeSpace($bytes){         //将字节转换为合适的单位
    $si_prefix = array( 'B', 'KB', 'MB', 'GB', 'TB', 'EB', 'ZB', 'YB' );
    $base = 1024;
    $class = min((int)log($bytes , $base) , count($si prefix) - 1);
    return sprintf('%1.2f' , $bytes / pow($base,$class)) . ' ' . $si
    prefix[$class] ;
}
?>
```

由于 disk_free_space()函数的返回值以字节为单位，因此上面自定义了一个 formatFreeSpace()函数将字节转换为合适的容量单位。执行后的输出结果如下：

```
网站所在磁盘的剩余空间为:65.29 GB
系统盘的剩余空间为: 7.01 GB
```

8.9.2　获取磁盘总容量

disk_total_space()函数语法与 disk_free_space()函数相同，语法格式如下所示：

```
float disk_total_space ( string $directory)
```

与 disk_free_space()函数不同的是，disk_total_space()函数以字节为单位返回指定目录

所在磁盘的总容量。

例如，要获取网站根目录所在磁盘和系统盘的总容量，可用如下代码：

```
<?php
    $drive="/";
    $freespace=disk_total_space($drive);
    echo "网站所在磁盘的总容量为:".formatFreeSpace($freespace);
    $cfree=disk_total_space("C:");
    echo "\n 系统盘的总容量为: ".formatFreeSpace($cfree);
?>
```

上述代码同样调用了自定义函数 formatFreeSpace()进行单位的转换。执行后的输出结果如下：

```
网站所在磁盘的总容量为:220.01 GB
系统盘的总容量为: 25.75 GB
```

8.9.3 获取目录占用空间

在 PHP 中提供了用于获取文件大小的 filesize()函数、获取磁盘可用空间的 disk_free_space()函数以及获取磁盘总容量的 disk_total_space()函数，但是并没有提供系统函数用于获取目录占用的空间。

这个功能对于站长来说非常重要。因为通常一个网站对应一个目录，为了掌握网站空间的使用情况，就必须获取目录的总容量大小。

由于没有系统函数可以使用，这里创建了一个自定义的 PHP 函数来完成这个任务。具体步骤如下所示。

（1）首先创建一个 PHP 页面，再创建一个名为 getDirSize()的自定义函数实现统计目录大小的功能。具体实现代码如下所示：

```
<?php
//计算目录大小的函数
function getDirSize($dir)
{
    //打开文件
    $handle = opendir($dir);
    //读取目录中的每个文件
    while (false!==($FolderOrFile = readdir($handle)))
    {
        //过滤掉某些未知的文件
        if($FolderOrFile != "." && $FolderOrFile != "..")
        {
            //确定文件大小并进行合并
            if(is_dir("$dir/$FolderOrFile")) {
                $sizeResult += getDirSize("$dir/$FolderOrFile");
```

```
                } else {
                    $sizeResult += filesize("$dir/$FolderOrFile");
                }
            }
        }
    //关闭文件目录
    closedir($handle);
    //返回目录大小
    return $sizeResult;
}
?>
```

从上述代码中可以看到，getDirSize()函数的实现并不复杂，而且都给出了注释，其中用到的大多函数在本章前面都有介绍，这里就不再重复。

（2）由于 PHP 的内置函数的返回值都是以字节为单位的。这里为了方便，以正常的容量单位显示，需要在页面添加 8.9.1 节中建立的 formatFreeSpace()函数。

（3）有了这两个函数作为基础，用于获取目录大小的功能就可以实现了。接下来编写代码调用函数进行测试，如下所示：

```
<?php
$mydir="D:/PHPROOT/htdocs";                    //要获取的目录
$size_zip=getDirSize($mydir);                  //计算出目录大小
$size_zip=formatFreeSpace($size_zip);          //转换单位
echo $mydir."目录的大小为: ".$size_zip;        //输出结果
$mydir="D:/PHPspace";                          //要获取的目录
$size_zip=getDirSize($mydir);                  //计算出目录大小
$size zip=formatFreeSpace($size zip);          //转换单位
echo $mydir."目录的大小为: ".$size zip;        //输出结果
?>
```

（4）保存 PHP 文件并执行，将会输出结果如下所示：

```
D:/PHPROOT/htdocs 目录的大小为: 1.50 MB
D:/PHPspace 目录的大小为: 48.02 MB
```

8.10 项目案例：简单文件管理系统

通过对本章的学习，读者掌握了如何在 PHP 中对文件进行操作，像新建、写入、移动和删除操作等。这些功能主要是靠 PHP 系统函数实现的，它不但提供了完整的文件操作，而且可以操作目录和磁盘。

这里将综合应用本章知识实现一个简单的文件管理系统，使读者能够很好地掌握管理文件和目录的各种知识。本案例最终的系统具有如下功能。

- **查看目录列表和文件详解。**

- 可以进入某个目录或者返回根目录。
- 对文本文件的写入、保存和删除。
- 对图片文件进行查看和删除。
- 创建文件和目录。

【实例分析】

在实现时首先需要为系统设置一个根目录，而且 PHP 程序应该有对该目录的完全访问权限。例如，使用 root 目录作为根目录，所有的操作都以它为基础，其中包含很多文件和子目录。

另外，在遍历目录时使用 scandir()函数实现，因为它比 readdir()函数效率高，而且无需关闭目录。需要注意的是，如果文件名或目录名中含有中文，则返回的是 GBK 编码。所以如果浏览器默认的是用 UTF-8 或者其他编码，则不能正常显示返回的值。

如果不想每次手动调整浏览器的编码设置，则应在 PHP 文件最开始添加如下代码：

```
header("Content-type: text/html; charset=GBK");
```

具体步骤如下。

（1）新建一个 index.php 文件作为系统的首页，并在同级目录新建 root 作为根目录。

（2）在页面的显著位置显示当前位于的目录，以及显示目录列的布局。最终代码如下所示：

```
<h2><a href="#">首页</a> &raquo; <a href="#" class="active">文件管理</a>
&raquo;
    <?php echo $pathname;?></h2>
<table cellpadding="3" cellspacing="3">
  <thead><tr>
    <th width="326" >名称</th>
    <th width="87" >类型</th>
    <th width="98" >大小</th>
    <th width="148" >操作</th>
  </tr>
</thead>
    <?php dirOutput($pathname); ?>
</table>
<a href="index.php">返回根目录</a>
```

在上述代码中，$pathname 变量保存的是目录，调用 dirOutPut()函数遍历$pathname 目录的内容。

（3）在页面的顶部添加如下代码，定义根目录和子目录。

```
<?php
$pathname="root";                    //定义根目录
if(isset($_GET["d"])){               //判断是否有子目录
    $pathname=$_GET["d"];            //获取子目录
}
```

```
?>
```

（4）创建 dirOutput()函数实现遍历指定目录，并以表格的形式输出，实现代码如下所示：

```
function dirOutput($pathname)       //以表格的形式遍历$pathname 目录中的内容
{
    if(!is dir($pathname)|| !is readable($pathname))
                                    //如果目录不存在就退出
        return null;
    $allFiles=scandir($pathname);   //获取目录内容的数组
    foreach ($allFiles as $filename) {          //遍历数组
        if(in_array($filename, array(".",".."))) continue; //排除特殊目录
        $fullname=$pathname."/".$filename;      //定义完整目录
        echo "<tr>";
        if(is_dir($fullname))                   //输出目录的内容
        {
            echo "<td><img src=style/img/ml.gif><a href=\"?d=$fullname\">"
            .$filename."</a></td>";
            echo "<td>目录</td>";
            echo "<td></td><td><a href=\"?d=$fullname\">打开</a></td>";
        }else
        {
            $pathinfo = pathinfo($filename);
            $td="<td></td>";
            $flag="<img src=style/img/wb.gif>";
            if($pathinfo['extension']=="txt")       //针对文本文件的输出
            {
                $td="<td><a href=\"ViewText.php?p=$fullname\">查看内容</a>
                | <a href=?a=d&p=$fullname>删除</a></td>";
            }elseif(in array($pathinfo['extension'], array("bmp","jpg",
            "png","gif")))
            {                                   //针对图片文件的输出
                $td= "<td><a href=\"ViewPic.php?p=$fullname\">查看图片</a>
                | <a href=?a=d&p=$fullname>删除</a></td>";
                $flag="<img src=style/img/tp.gif>";
            }
            echo "<td>$flag".$filename."</td>";
            echo "<td>".$pathinfo['extension']."类型</td>";
            echo "<td>".formatFreeSpace(filesize($fullname))."</td>";
            echo $td;
        }
        echo "</tr>";
    }
}
```

从上述代码中可以看出，针对目录、文本文件和图片文件有不同的输出。运行 index.php 将看到如图 8-12 所示的首页运行效果。

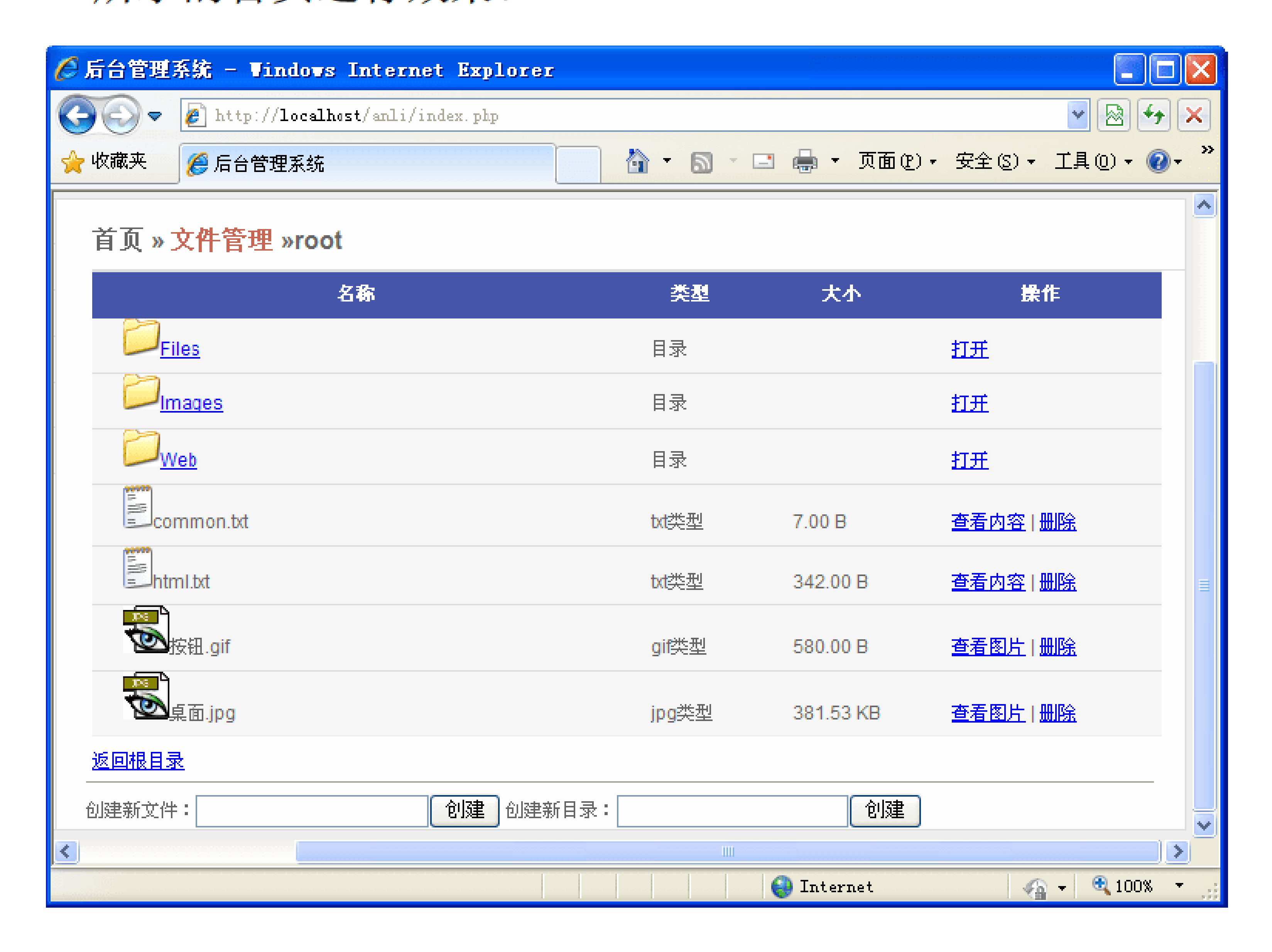


图 8-12　文件管理系统首页效果

（5）在系统首页中单击目录名称或者后面的“打开”链接可以查看该目录下的内容。图 8-13 所示为查看 root/Files 目录时的效果。

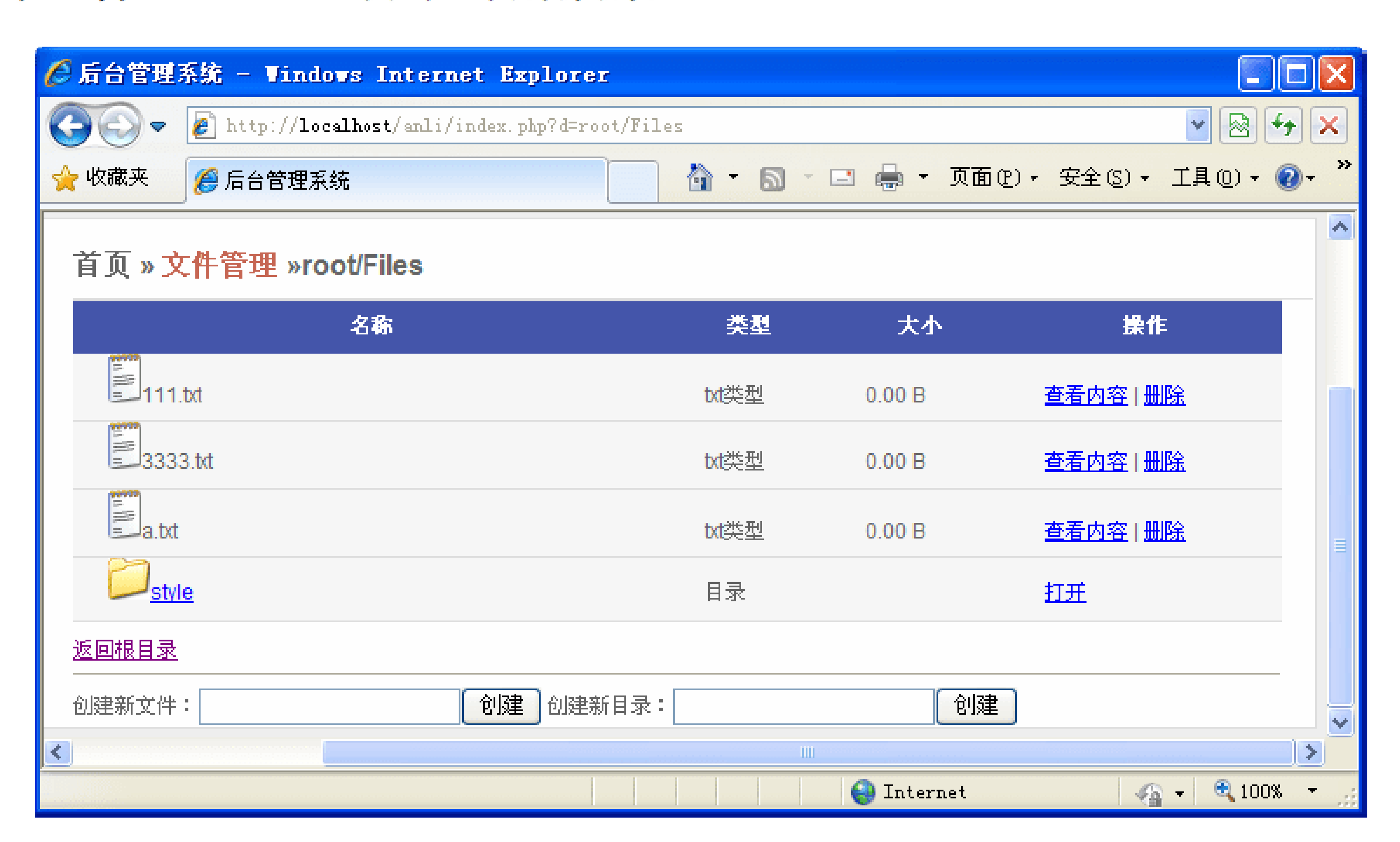


图 8-13　查看 root/Files 目录内容

（6）无论位于哪个目录，在页面的底部都会显示创建新文件和目录的表单，如图 8-12 和 8-13 所示。如下所示为该表单的代码：

```
<form action="" method="POST">
创建新文件：<input name="newfile" type="text" /><input name="newf"
type="submit" value="创建" id="newf" />
创建新目录：<input name="newdir" type="text" /><input name="newd" type=
"submit" value="创建" id="newd" />
```

```
</form>
```

（7）如下所示为创建文件和目录的 PHP 端接收代码：

```
if(isset($_POST["newf"])){
    createNewFile($_POST["newf"]);  //调用 createNewFile()函数创建新文件
}
if(isset($_POST["newd"])){
    createNewDir($_POST["newd"]);   //调用 createNewDir()函数创建新目录
}
```

（8）为了实现创建新文件和目录，这里自定义了 createNewFile()和 createNewDir()两个函数。如下所示是这两个函数的实现代码：

```
function createNewFile($filename){      //创建$_filename 指定的新文件
    $pathname=isset($_GET["d"])?$_GET["d"]:"root";
    if(file_exists($pathname."/".$filename))    {
        echo "<script>alert('要创建的文件已存在');</script>";
    }
    else{
        $handle = fopen($pathname."/".$filename, 'w');
        fclose($handle);
    }
}
function createNewDir($dirname) {           //创建$dirname 指定的新目录
    $dirname=$_POST["newdir"];
    $pathname=isset($_GET["d"])?$_GET["d"]:"root";
    $path=$pathname."/".$dirname;
     if(!is_readable($path)){
        is_file($path) or mkdir($path,0700);
    }
}
```

（9）接下来编写用于删除文本文件和图片文件的代码。单击这两种链接之后都会在 URL 中添加一个 a 参数，接收代码如下所示：

```
if(isset($_GET["a"])&&isset($_GET["p"])){       //判断是否删除动作
    deleteFile($_GET["p"]);                 //调用删除函数 deleteFile()
}
function deleteFile($filename)              //删除$filename 指定的文件
{
    if (file_exists($filename)) {           //判断文件是否存在
        if(unlink($filename)) {             //执行删除操作
            echo "<script>alert('文件删除成功');</script>";   //删除成功
        }
        else {
            echo "<script>alert('文件删除失败');</script>";   //删除失败
```

```
            }
        }
    }
```

（10）经过上面步骤的编码，系统的首页就制作完成了。可以进入某个目录，创建新文件或者目录，也可以删除文件。下面编写针对文本文件的读取和保存代码。

当在首页中单击“查看内容”链接时会转到 ViewText.php 文件。在该文件中不仅可以查看文件的原始内容，还可以进行修改并保存。核心实现代码如下所示：

```
     <h2><a href="#">首页</a> &raquo; <a href="#" class="active">文件管理</a>
     &raquo; 查看文件内容</h2>
      <?php
     if(isset($_POST["save"])){
        $str=$_POST["filetext"];
        $fname=$_POST["fname"];
        $s=file_put_contents($fname,$str);
        if($s)  {echo "<script>alert('保存成功.');</script>";}
    }
    if(isset($_GET["p"])){
        $filename=$_GET["p"];
        if(file_exists($filename))
        {
    ?>
          <form id="form1" name="form1" method="POST" action="viewtext.php">
            <input type="hidden" name="fname" id="hiddenField" value="<?php
            echo $filename; ?>" />
            <table cellpadding="3" cellspacing="3">
              <thead>
                <tr>
                  <th>文件名称: <?php echo $filename;?></th>
                </tr>
              </thead>
              <tr>
                <td><?php $str=file_get_contents($filename);  ?>
        <textarea name="filetext" id="textarea" cols="90" rows="10"><?php echo
        $str; ?>  </textarea>
                  <br />
                  <input type="submit" name="save" id="save" value="保存" />
                  </td>
              </tr>
            </table>
          </form>
          <?php
    }}
    ?>
```

（11）从首页中单击文本文件的“查看内容”链接进入 ViewText.php 页面。图 8-14 所示为对 root/html.txt 文件进行操作时的效果。

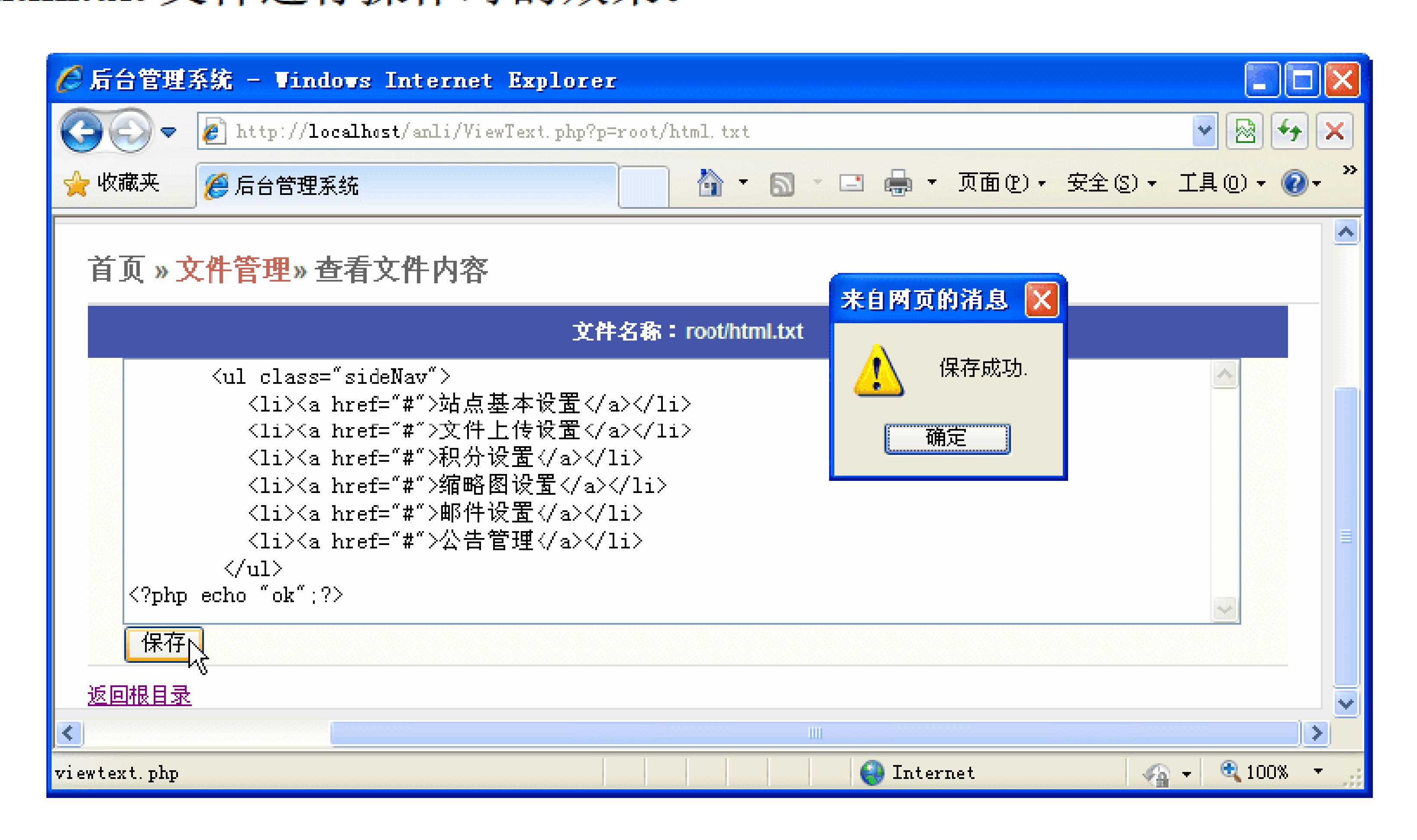


图 8-14　文本文件操作

（12）在首页中如果单击“查看图片”链接将会在 ViewPic.php 中浏览图片，如图 8-15 所示。

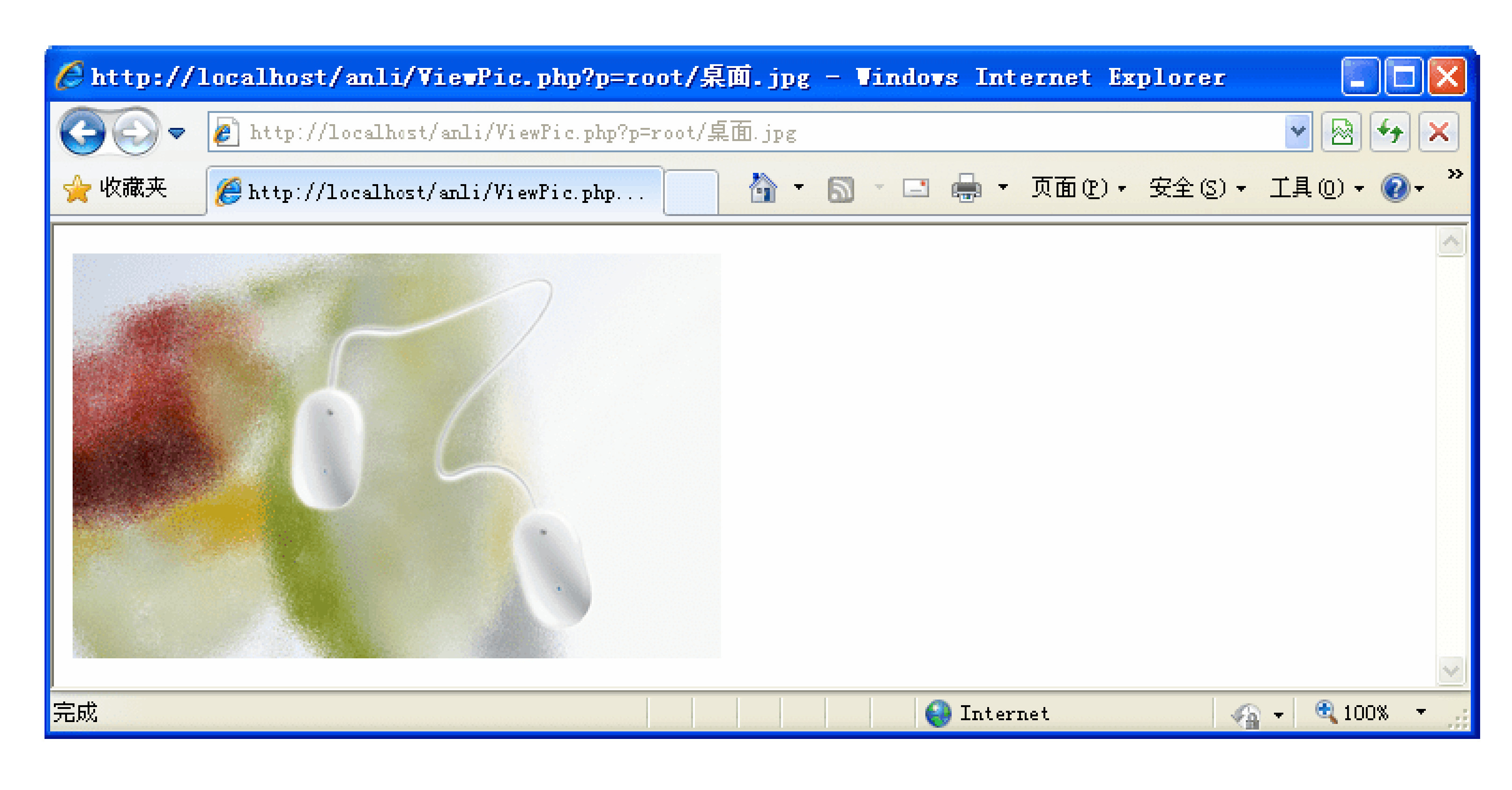


图 8-15　查看图片文件

（13）ViewPic.php 文件的实现比较简单，它会根据传递的 p 参数来读取图片数据并显示，实现代码如下所示：

```
<?php
if(isset($_GET["p"])){                              //判断是否为查看图片动作
    $filename=$_GET["p"];                           //获取图片文件名
    if(file_exists($filename))  {                   //判断文件是否存在
        $fp = fopen($filename, 'rb');               //以二进制打开方式
        header("Content-Type:image/gif");           //设置图片的类型
        header("Content-Length:" . filesize($filename));//设置图片的大小
        fpassthru($fp);                             //将图片的二进制流输出到页面
        exit;                                       //终止程序
    }else{
```

```
        echo "指定的文件不存在。";
        }
}else{
    echo "<script>window.location='index.php';</script>";
}
?>
```


8.11　习题

一、填空题

（1）在 PHP 使用__________函数可以获取文件最后一次修改的时间戳。
（2）在 fstat()函数返回的数组中__________键表示文件大小。
（3）在下面代码的空白处填写代码实现按行遍历文件。

```
  <?php
    $filename = "C:\test.txt";                    //定义文件名
    $lines = __________ ($filename);              //将一个文件读入数组$lines
    foreach ($lines as $line_num => $line) {
        echo "[{$line_num}] : " .$line . "</li>";
    }
    ?>
```

（4）假设有下面一段 PHP 代码，执行后程序输出的结果是__________。

```
<?php
    $path="www.itzcn.com/index.html";
    $suffix=".html ";
    echo basename($path,$suffix);
?>
```

（5）遍历目录的__________函数执行后会返回包含文件和目录的数组。
（6）如果要获取某个磁盘的可用空间应该使用__________函数。

二、选择题

（1）filetype 函数返回____________时表示是一个目录。

A．dir

B．disk

C．block

D．driver

（2）下面打开文件的代码使用错误的是____________。

A．fopen("musicList.txt","r")

B．fopen("C:\musicList.txt","x")

C．fopen("ftp://musicList.txt","w+")

D．fopen("C:/../\mud\sicList.txt","r")

（3）下面的哪个函数不可以按行读取文件内容____________。

A．file()

B．fread()

C．fgets()

D．fgetcsv()

（4）假设 test.txt 文件的内容如下：

```
ABCDEFG
abcdefg
```

下面代码的输出结果是____________。

```
<?php
$filename=" test.txt";
$file = fopen($filename,"r");
echo fgets($fp);
fseek($file,0);
echo fgetc($file);
fclose($file);
?>
```

A．ABCDEFG

B．abcdefg

C．ABCDEFGabcdefg

D．ABCDEFGa

（5）下面不属于目录操作函数的是__________。

A．opendir

B．mkdir

C．copy

D．rmdir

（6）pathinfo()函数返回的数组中，__________键表示路径中文件的扩展名。

A．dirname

B．basename

C．filename

D．extension

三、上机练习

1．查看列车信息

假设在 files 目录下有一个 trian.txt 文件，它保存了列车信息，包括车次、始发站和终

点站，每个信息之间使用分号分隔，每行显示一条。例如，其中的内容如下所示：

```
K234;上海;石家庄
K920;汉口;天津
T146;南昌;北京西
T10;重庆;北京西
```

现在要求读取 trian.txt 文件，并在页面上显示所有列车信息，最终效果如图 8-16 所示。

车次	始发站	终点站
K234	上海	石家庄
K920	汉口	天津
T146	南昌	北京西
T10	重庆	北京西

图 8-16　运行效果

2. 操作文件和目录

根据本章所学习的 PHP 对文件和目录操作的知识，完成如下练习：

（1）在网站根目录下新建 document/doc 目录。
（2）将上机实践 1 的 train.txt 文件复制到 doc 目录下。
（3）将 train.txt 重命名为 test.txt。
（4）向 test.txt 文件的末尾追加一行，并输出写入的字节数。
（5）使用按行读取函数输出 test.txt 文件的内容。
（6）显示 test.txt 文件的大小、创建时间和上次修改时间。

8.12　实践疑难解答

8.12.1　删除目录及目录下所有文件的问题



删除目录及目录下所有文件的问题
网络课堂：http://bbs.itzcn.com/thread-19692-1-1.html

【问题描述】：请看如下的 PHP 代码

```
/**
* 删除目录及目录下面的所有文件
* @param    string    $dir        路径
* @return    bool    如果成功则返回 TRUE，失败则返回 FALSE
*/
function dir delete($dir) {
    $dir = dir path($dir);
    if (!is_dir($dir)) return FALSE;
    $list = glob($dir.'*');
```

```
        foreach($list as $v) {
            is dir($v) ? dir delete($v) : @unlink($v);
        }
        return @rmdir($dir);
    }
```


以上是 PHPCMS 使用到的代码。我想问下为何这么做，直接删除$dir 不可以么？为何还要遍历其下面的所有的目录，然后再删除？

【解决办法】：

首先要知道在 PHP 中删除目录使用的是 rmdir()函数，而该函数有一个要求就是目录必须为空才能删除。如果要删除的目录中有文件和目录，则可能造成删除失败。

因此，开发人员出于安全的考虑在删除目录时对其中的内容进行遍历。如果还有目录则循环删除。

8.12.2　如何递归遍历一个文件夹下面的层次目录



如何递归遍历一个文件夹下面的层次目录

网络课堂：http://bbs.itzcn.com/thread-19693-1-1.html

【问题描述】：如题，用 PHP 递归遍历一个文件夹下面的层次目录，输出方式按照数组输出文件名。

我写的代码如下，这个不能实现所要的功能，不能体现出层次的结构。

```
<?php
//遍历文件夹目录及文件
function get_all_files($path) {
    $list = array();
    foreach(glob($path . '/*') as $item) {
        //如果是文件
        if (is dir($item)) {
            $list[basename($item)] = array merge($list , get all files
            ($item));
        } else {
            //如果是目录
            $list[] = $item;
        }
    }
    return $list;
}
var_dump(get_all_files('d:/wenjianjia'));
?>
```

求实现的思路，最好有代码，谢谢。

【解决办法】：实现这个功能，最简单的方法可能就是用递归了。先说一下思路：首先是判断根目录下的所有文件和目录，这个是单层次遍历并不难。

然后用 foreach 语句进行遍历（当然也可以是其他循环语句），在遍历时分两种情况。第一种当判断是文件时就显示一下文件名，或者进行一步操作；第二种当判断是目录时，就递归函数本身，并将目录名称作为参数传递。

下面是以前项目中的递归代码，用的是 while 循环。

```
<?php
    function getAllFiles ($path) {
        $handle = @opendir($path);
        $handle file = "";
        while($handle file = readdir($handle)){
            if($handle file!=="." && $handle file!==".."){
                $handle files = $path."/".$handle file;
                if(is_dir($handle_files)){
                    echo "$handle_files <br />";
                    getAllFiles($handle files);
                }
                if(is file($handle files))
                    echo "$handle files <br />";
            }
        }
        closedir($handle);
    }
getAllFiles('wwwroot/uploadfiles');
?>
```

第9章 与 Web 页面交互

要开发 Web 网站就不能不提到 HTML，它是 Web 页面设计的标准，因此了解 HTML 是必须的。在 HTML 中的表单为浏览者提供了一个与网站进行互动的平台，它可以将数据传递到 PHP。PHP 也可获取表单的数据并输出。

除此之外，在网站的多个页面之间如何存储用户的相关信息，也是 Web 开发时的重点。在 PHP 中可以通过 Cookie 和 Session 机制来实现。

本章将详细介绍 Web 开发时表单如何与 PHP 进行结合，像获取表单数据、遍历表单、动态生成表单等，还介绍了常见的表单处理技巧。同时还介绍了 Cookie 和 Session 的使用，最后实现了文件的上传和下载功能。

本章学习要点：

- 了解表单的组成元素以及与 PHP 的关系
- 掌握获取表单数据的方法
- 掌握表单常见操作和处理技巧
- 熟悉 URL 中编码和解码操作
- 掌握 Cookie 的使用
- 掌握 Session 的使用
- 掌握 PHP 实现文件上传与下载的方法

9.1 表单

表单在网页中主要负责数据采集功能。一个表单有 3 个基本组成部分：表单标签、表单域和表单按钮。

表单标签包含处理表单数据所用程序的 URL 以及数据提交到服务器的方法。表单域包含让用户选择和输入的区域，像文本框、密码框和复选框等。表单按钮一般用于将数据提交到服务器进行处理，包括提交按钮、复位按钮和一般按钮。

9.1.1 表单与 HTML

表单在 HTML 中由<form>标记定义，它是 HTML 的一个重要组成部分，主要用于搜集不同类型的用户输入和数据传递。在 HTML 表单中包含了很多表单元素，通过它们允许用户单击、选择和输入信息。

HTML 表单的创建语法如下所示：

```
<form name="form_name" method="method" action="url" enctype="value"
target="target" id="id">
  <!-- 此处放表单元素，这里省略  -->
</form >
```

在上述代码中，表单各个属性的说明如表 9-1 所示。

表 9-1　表单属性说明

属性名称	说明
name	表单的名称
method	设置表单的提交方式，有 GET 和 POST 两种
action	指向处理该表单页面的 URL，可以是相对位置或者绝对位置
enctype	设置表单内容的编码方式
target	设置返回信息的显示方式，可选值有_blank、_parent、_self 和_top
id	表单的 ID 号

从语法格式中可以看出，在 form 中出现的内容称为表单元素，表 9-2 中列出了可以包含的表单元素信息。

表 9-2　form 表单元素

元素名称	说明
button	表示按钮的布局控件对象
checkbox	表示复选框字段对象
fileupload	表示文件上传表单字段的对象
hidden	表示表单的隐藏域对象
password	表示密码输入框控件对象
radio	表示单选按钮对象
reset	表示复位按钮对象
select	表示下拉列表对象
submit	表示提交按钮对象
text	表示文本输入框控件对象
textarea	表示多文本输入区域的对象

图 9-1 所示的效果使用了表 9-2 中的表单元素，其中的大部分表单元素我们都很熟悉。例如，使用 input 定义表单中的单行输入文本框、输入密码框、单行按钮、复选框、隐藏控件、重置按钮及提交按钮；使用 select 在表单定义下拉菜单和列表框；使用 textarea 在表单中创建多行文本框（文本区域）；等等。

9.1.2　表单与 PHP

对于大多数基于数据库的网站，网站的开发人员需要处理很多的事情。这不仅包括从数据库读取数据动态生成网站，为了使网站具有交互性还需要做额外的工作，即使只有一个搜索框。

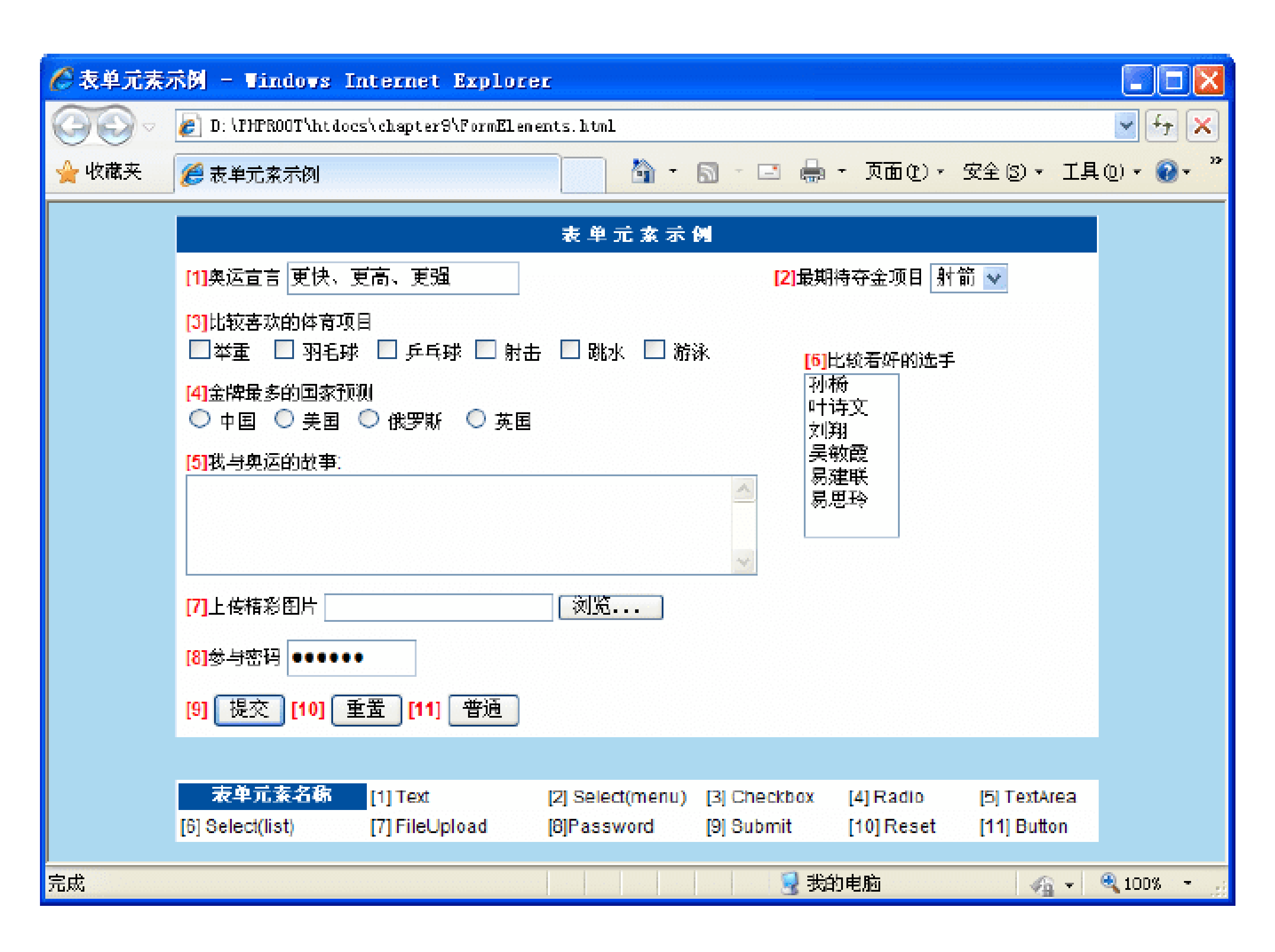


图 9-1　使用 form 表单元素

在网站开发时，JavaScript 主要用于前端页面与用户的交互处理。利用它可以直接响应用户的行为，例如动态提示、数据验证等。而 PHP 主要负责在服务器端收到请求的数据，处理后发送给客户端浏览器响应。

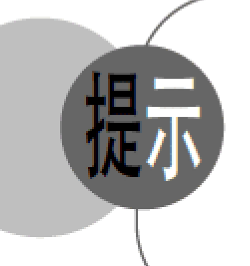
那么究竟 PHP 如何与表单进行交互呢？下面通过具体的实例讲解 PHP 为表单元素赋值的方法。例如，对表单元素的文本框进行赋值，只需要将所赋的值添加到文本框所对应的 value 属性中即可。代码如下：

```
<?php
   $name="校园杯唱歌比赛";        //为变量$name 赋值
?>
标题: <input name="name" type="text" id="name" value="<?php echo $name;?>"
/>
```

例如，对表单元素隐藏域进行赋值，只需要将所赋的值添加到 value 属性后即可。代码如下：

```
<?php
 $id="HNZZ04410474";  //为变量$id 赋值
?>
隐藏域 HID 的值:<input  type="hidden" name="HID"  value="<?php echo $id;?>" >
```

从上面代码中可以看出，首先为变量$id 赋初始值，然后将变量$id 的值赋给隐藏域。



在程序开发过程中经常用隐藏域来存储一些不必要显示的信息或需要传送的参数。隐藏域的值在程序运行过程中并不可见，为了看到效果，可以通过 echo 语句将隐藏域的 name 属性值进行输出，如"echo $_POST["HID"];"。

【实践案例 9-1】

假设在一个留言表单中包括姓名、邮箱和内容 3 项。下面使用 PHP 为这 3 项赋值，使表单一打开其中表单元素就有值，具体步骤如下。

（1）将有留言表单的页面另存为 index.php。因为这里要添加 PHP 脚本，而 HTML 不能执行。

（2）在页面定位到留言表单 form 的上方。再添加如下的代码创建一些变量并赋值：

```
<?php
$nameTip="在这里留下你的姓名";
$emailTip="留下你的邮箱方便我与您联系";
$textTip="有什么要说的，写到这里";
?>
```

（3）在留言表单中对输入姓名的 input 元素进行修改，在这里使用上面的$nameTip 变量。代码如下：

```
<label for="posName">姓名 <span>(必须)</span></label><br>
<input name="posName" type="text" class="text" id="posName" value="
<?php echo $nameTip; ?>" size="25">
```

（4）然后对输入邮箱的 input 元素进行修改，在这里使用上面的$emailTip 变量。代码如下：

```
<label for="posEmail">邮箱 <span>(必须)</span></label><br>
<input class="text" size="25" name="posEmail" id="posEmail" type="text"
value="<?php echo $emailTip; ?>">
```

（5）最后对输入留言内容的 textarea 元素进行修改，在这里使用上面的$textTip 变量。代码如下：

```
<label for="posText">内容 <span>(必须)</span></label><br>
<textarea cols="50" rows="3" name="posText" id="posText"><?php echo
$textTip; ?></textarea>
```

（6）完成上述修改步骤后，保存对 index.php 的修改然后在浏览器中查看效果。此时，页面打开后将会在表单中显示指定的内容，如图 9-2 所示。

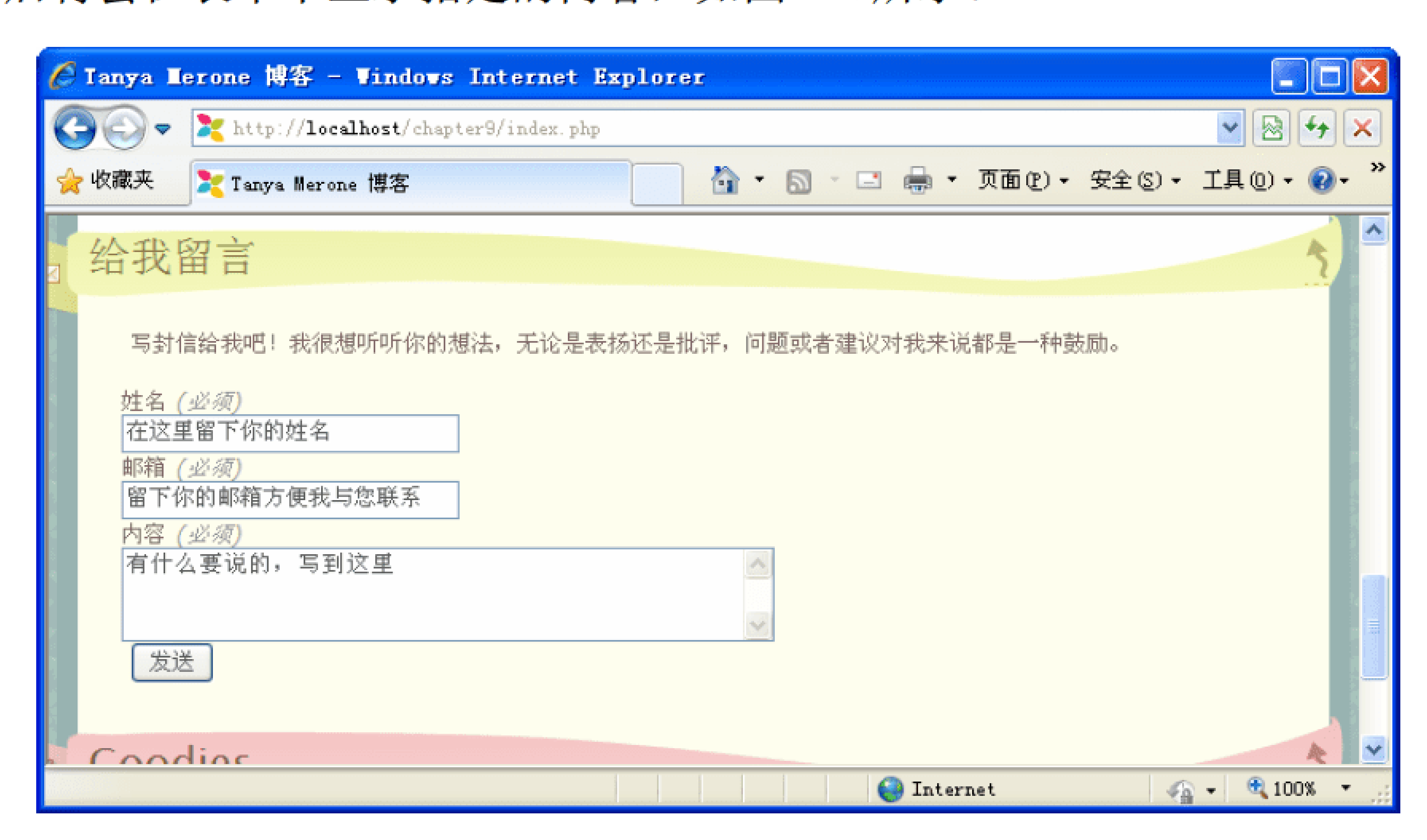


图 9-2 为留言表单赋值

9.2 获取表单数据

表单是 Web 应用中最常用的功能之一，由文本输入框、下拉菜单、复选框和按钮等元素组成。表单用于用户输入和提交信息，常见的有注册用户、发表留言、添加商品等交互功能，表单填写完毕后提交给服务器端的 PHP 处理。本节将详细介绍如何在 PHP 中获取由表单提交的数据。

9.2.1 设置表单提交方式

HTML 表单（Form）有两种提交方式：GET 和 POST，采用哪种方式提交表单数据由 form 元素的 method 属性值决定。这两种方式在数据传输过程中分别对应 HTTP 协议的 GET 和 POST 方法。

1. GET 方式

GET 是表单的默认提交方法，使用浏览器地址栏来传递数据。例如，如下所示是在 baidu 和 google 上搜索 hello 关键字后转到的 URL 地址。

```
http://www.baidu.com/s?wd=hello&rsv bp=0&rsv spt=3&rsv n=2&inputT=944
https://www.google.com.hk/search?source=ig&hl=zh-CN&rlz=1G1GGLQ&q=hello
&btnG=Google+%E6%90%9C%E7%B4%A2
```

可以看到地址中间问号之后的字符串，例如“wd=hello”、“q=hello”、“btnG=Google+%E6%90%9C%E7%B4%A2”等这些称为参数，使用的是“参数名=参数值”的形式，多个参数之间使用“&”分隔。在问号之前的是接收参数的 URL 地址。整行的作用是将问号后面的各个值传递到前面的 URL 地址进行处理，该地址由表单的 action 属性指定。

下面代码创建一个使用 GET 方式向 server.php 提交的表单：

```
<form action="server.php" method="get" id="userform">
</form>
```

提交之后将会看到类似如下的 URL 地址：

```
Server.php?k=hello&n=bbs&type=diary&charset=utf-8&type=diary
```



对于 GET 方式提交的数据可以由 PHP 中的$_GET 数组进行接收，将在 9.2.2 节介绍。

GET 方式的优点是方便直观、通过表单输入或者网页超链接就可以实现。缺点就是不安全，因为在传输的过程中，数据被放在请求的 URL 中，这样就可能会有一些隐私的信息

被第三方看到。用户也可以在浏览器中直接输入 URL 提交数据。而且，受 URL 长度的限制，GET 方式能传输的数据也比较小。

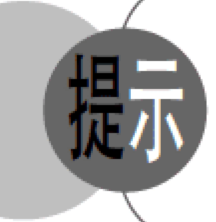
另外，当 GET 方式中传递的字符串中含有汉字或者其他非 ASCII 字符时，需要额外的编码转换。

2. POST 方式

使用 POST 方式提交时，数据将随着页面请求的 HTTP 数据包一起发送，因为不会在 URL 地址栏上体现，因此用户只会看到提交后的地址，而不会看到其他数据。

下面代码创建一个使用 POST 方式向 server.php 提交的表单：

```
<form action="server.php" method="post" id="userform">
</form>
```



对于 POST 方式提交的数据可以由 PHP 中的$_POST 数组进行接收，将在 9.2.3 节介绍。

当用 POST 方法提交数据时，整个过程是不透明的。这是因为数据会附加到 HTTP 协议的 header 信息中，用户也不能随意修改。这一点对于网站来说，安全性要比 GET 强，而且也可以发送大体积的数据。

因为 POST 是随 HTTP 的 header 信息一起发送的，因此一旦 POST 表单提交，如果用户使用浏览器的“后退”按钮，浏览器不会重新提交 POST 数据。但是，如果此时单击“刷新”按钮，将会有“数据已经过期，是否重新提交表单”的提示，这一点没有 GET 方便。在 GET 提交时，无论用户使用后退还是刷新功能，浏览器的 URL 地址仍然存在。

POST 与 GET 的另一个不同点是，当对 GET 方式提交后的页面使用收藏夹，下次再访问的时候会直接进入这个页面。而如果是收藏 POST 方式提交后的页面，再次访问时可能没有任何数据，因为缺少提交值。这也是搜索引擎网站使用 GET 方式处理关键字的原因。

9.2.2 获取 GET 提交的数据

对于 GET 方式提交的表单，在 PHP 页面使用$_GET 变量可以提交的数据。$_GET 变量其实是一个数组，用于获取来自 method="get"的表单中的值，也可以获取在 URL 地址中问号后面的参数值。

下面通过一个简单的示例，演示如何直接从 URL 地址栏中获取数据。假设，在一个 PHP 页面中有如下代码：

```
<?php
  $start=$_GET["start"];          //获取 URL 地址栏中参数名称为 start 的值
  $end=$_GET["end"];              //获取 URL 地址栏中参数名称为 end 的值
  echo "查询条件: 从<b>".$start."</b> 到 <b>".$end."</b> 的数据";
?>
```

将页面保存为 admin.php。然后在浏览器中运行并在 URL 中使用如下形式指定参数值：

```
admin.php?start=值 1&end=值 2
```

例如，这里使用的是“admin.php?start=100&end=1000”，运行之后将看到如图 9-3 所示的效果。

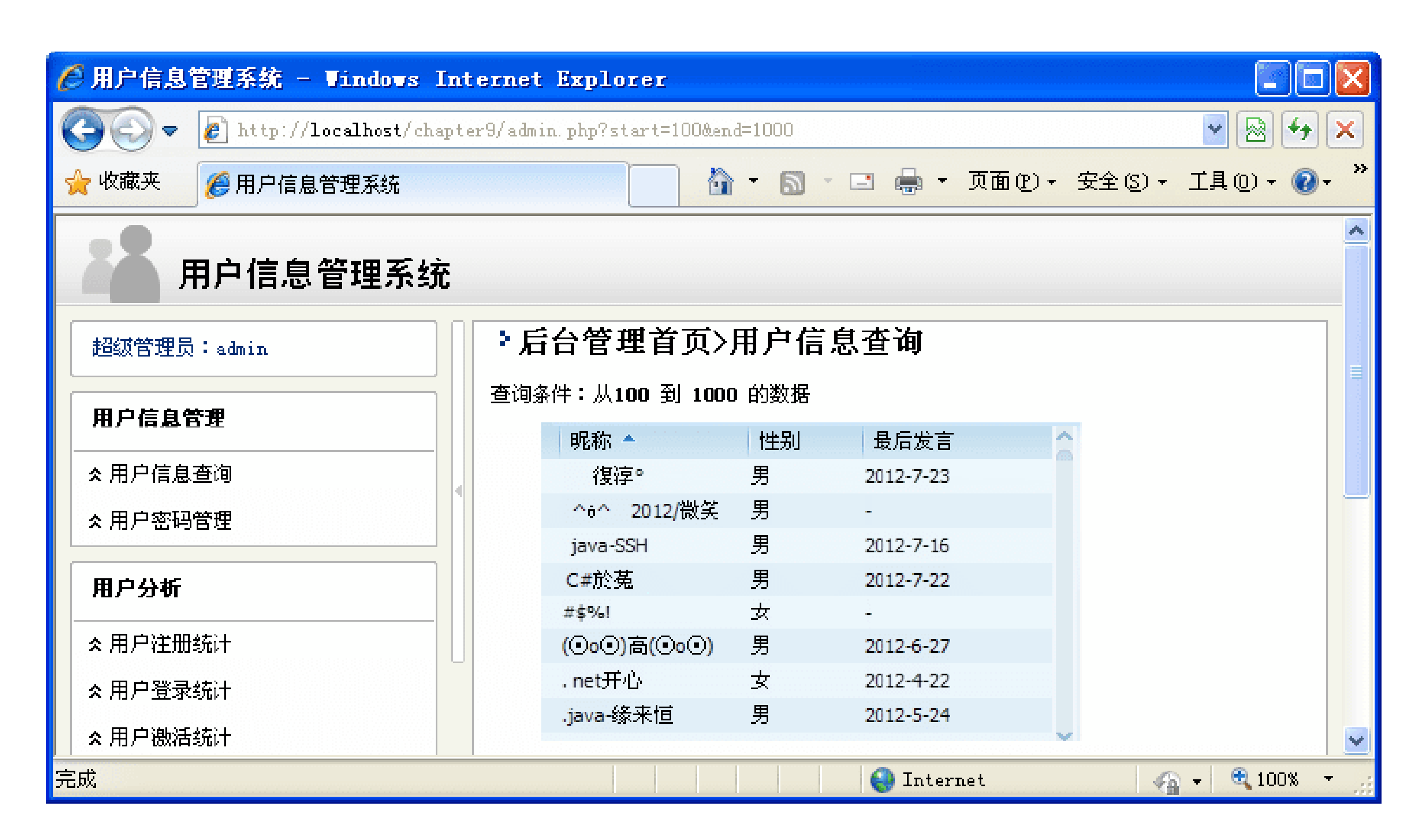


图 9-3　直接获取 URL 地址栏中的数据

在 PHP 中$_GET 是一个关联数组，它会自动包含在 URL 中问号后面查询字符串中的所有值。因此在查询字符串中传递变量名称可以访问，例如本示例中的$_GET["start"]获取了 start 参数的值。

通过本示例可以看到，PHP 的使用非常简单，只用一行代码就可以获取 URL 中的参数。但是上面的示例有一个安全隐患，例如使用如下的地址进行提交：

```
admin.php?start=<h1>hello</h1>&end=<h1>PHP</h1>
```

这里为 start 参数和 end 参数指定的均是 HTML 代码，再次运行后将会看到这两个参数值作为 HTML 代码显示，如图 9-4 所示。这是因为在使用$_GET["start"]获取到“<h1>hello</h1>”字符串之后并没有进行任何处理，而是直接显示。如果恶意用户使用复杂的代码可能会窃取用户信息，或者造成其他安全问题。

解决的办法很简单，就是将获取的值作为纯文本处理然后再显示。在 PHP 中调用 htmlspecialchars()函数即可转义其中的恶意标记。例如，使用如下代码进行替换后，再次运行将看到如图 9-5 所示的效果。

```
$start=htmlspecialchars($_GET["start"]);
$end=htmlspecialchars($_GET["end"]);
```

【实践案例 9-2】

以 9.1.2 节的留言表单为例，使用 GET 方式获取用户输入的留言信息，具体步骤如下。

（1）新建一个 PHP 页面，保存为 index_get.php。

（2）使用 form 在页面中制作留言表单，包括姓名、邮箱和内容。代码如下所示：

图 9-4　传递恶意代码效果

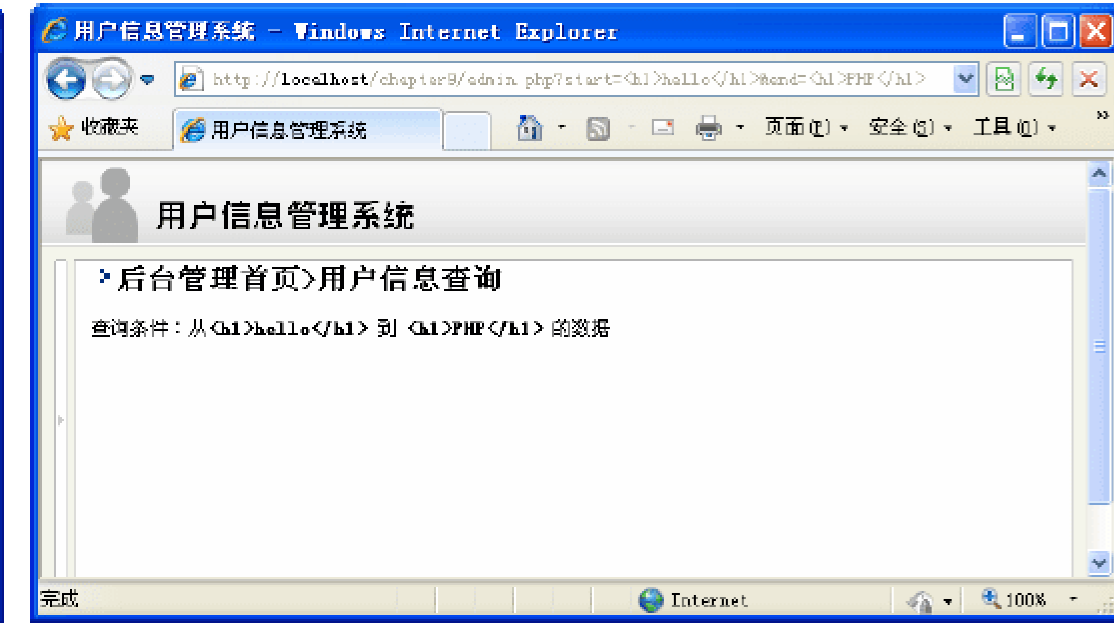


图 9-5　处理恶意代码效果

```
<form action="index get.php" method="get" id="msgForm">
   <label for="msgName">姓名 <span>(必须)</span></label>  <br>
   <input name="msgName" type="text" class="text" id="msgName" value=""
   size25">  <br>
   <label for="msgEmail">邮箱 <span>(必须)</span></label>       <br>
   <input class="text" size="25" name="msgEmail" id="msgEmail" typetext"
  value="">  <br>
   <label for="msgText">内容 <span>(必须)</span></label>       <br>
   <textarea cols="50" rows="3" name="msgText" id="msgText"></textarea>
   <br>
   <p><input class="button" name="sendmsg" id="sendmsg" value="发送"
   typesubmit">  </p>
</form>
```

form 标记的 action 属性指定表单提交地址为本页面，method 属性指定表单使用 get 方式提交。

（3）在评论表单下方编写 PHP 代码，实现使用$_GET 获取表单中的数据并输出，具体代码如下所示：

```
<?php
if(isset($_GET["sendmsg"]))                        //是否单击了"发送"按钮
{
    $name=$_GET['msgName'];                        //保存姓名
    $email=$_GET['msgEmail'];                      //保存邮箱
    $text=$_GET['msgText'];                        //保存内容
    echo("<b>$name</b> [$email]留言说: <br/>");
    echo("内容:$text");                            //输出留言内容
}
?>
```

由于“发送”按钮的 name 属性是 sendmsg，所以上述代码通过判断$_GET 数组中是否有 sendmsg 键来检测用户是否单击了“发送”按钮。

（4）在浏览器中查看页面运行效果。打开 index_get.php 文件，输入信息后单击“提交”按钮查看结果，如图 9-6 所示。

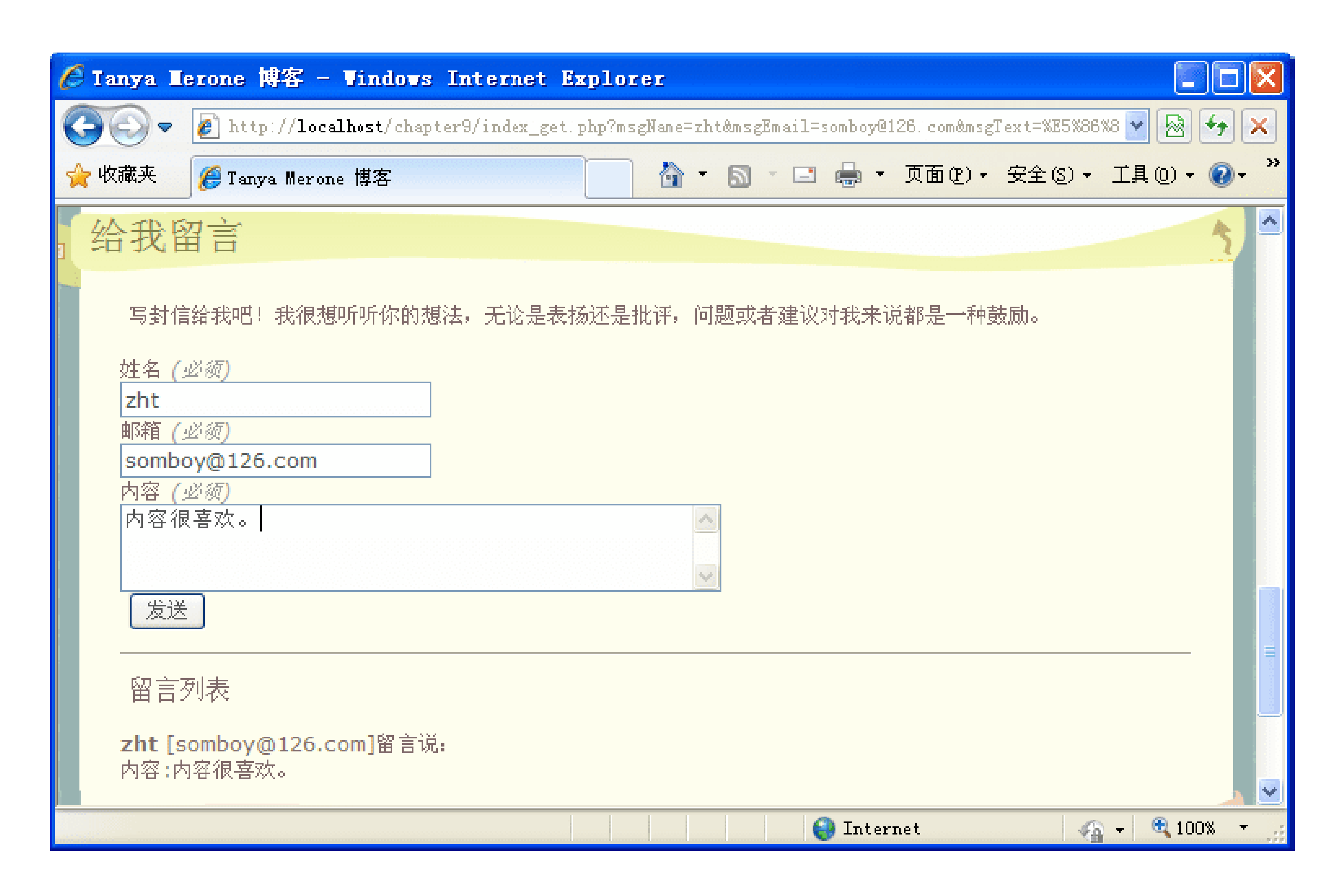


图 9-6　GET 方式查看留言

在如图 9-6 所示浏览器的地址栏中会看到传递的参数信息，如下所示：

```
http://localhost/chapter9/index_get.php?msgName=zht&msgEmail=somboy@126
.com&msgText=%E5%86%85%E5%AE%B9%E5%BE%88%E5%96%9C%E6%AC%A2%E3%80%82&sen
dmsg=%E5%8F%91%E9%80%81
```

由此信息可以看到，使用 GET 方式提交表单中的数据时，URL 和表单元素之间用“?”隔开，而多个表单元素之间用“&”隔开，每个表单元素的格式都是“name=value”，固定不变。

9.2.3　获取 POST 提交的数据

与 GET 方式相比，POST 方式提交后的 URL 地址非常简洁，这是因为 POST 数据是随 HTTP 请求一起发送的，因此不会在 URL 地址栏中看到。

对于 POST 方式提交的表单，在 PHP 中可以通过$_POST 变量来获取，它也是一个数组。其中的每个键对应表单中的一个元素。例如，表单中包含一个 name 为“username”的文本输入框，在使用 POST 方式提交数据后，PHP 可以使用$_POST["username"]获取用户输入的值。

【实践案例 9-3】

下面以实践案例 9-2 的留言表单为单，这里使用 POST 方式进行提交实现相同的功能。

（1）将 index_get.php 另存为 index_post.php。

（2）在 index_post.php 中将表单设置为 POST 提交，修改后的代码如下：

```
<form action="index_post.php" method="POST" id="msgForm">
```

（3）然后修改原来使用$_GET 方式获取数据有代码，改为$_POST 进行接收，修改后的代码如下所示：

```
<?php
```

```
if(isset($_POST["sendmsg"]))                        //是否单击了"发送"按钮
{
    $name=$_POST['msgName'];                        //保存姓名
    $email=$_POST['msgEmail'];                      //保存邮箱
    $text=$_POST['msgText'];                        //保存内容
    echo("<b>$name</b> [$email]留言说: <br/>");
    echo("内容:$text");                             //输出评论内容
}
?>
```

从上述代码中可以看出，$_POST 也是一个数组，用于收集来自 method="post"的表单中的值，由 HTTP POST 方式发送的变量名称和值组成。

（4）运行 index_post.php 页面，会发现表单与 GET 方式相同。输入内容之后单击“发送”按钮同样可以在下方看到留言，结果如图 9-7 所示，但是浏览器的 URL 不会发生变化。



图 9-7　POST 方式查看留言

9.3　表单的常见操作

通过 9.2 节的学习，我们了解了表单两种方式提交的区别以及如何获取提交的数据。本节将介绍一些网站开发时经常用到的表单操作及处理方式。

9.3.1　遍历表单

无论表单使用 POST 还是 GET 方式进行提交，在 PHP 中都是通过数组获取的。因此，可以使用 foreach 语句遍历数组中的所有元素，从而输出表单中元素的键和值。

具体的方法也很简单，首先确定要遍历的表单是以 POST 还是 GET 提交的。例如，这里的表单代码如下：

```
<form action="forEachForm.php" method="POST">
<!-- 此处省略表单元素的代码 -->
<input type="submit" name="submit" id="button" value="提交" />
</form>
```

如果要遍历上面的表单应该使用$_POST 数组，具体代码如下所示：

```
<?php
if(isset($_POST["submit"]))                //判断是否单击"提交"按钮
{
?>
     <table cellpadding="5" cellspacing="1">
       <tr height="20">
         <th colspan="2">表单中变量的值</th>
       </tr>
     <?php  foreach($_POST as $name => $value){ //遍历$_POST 数组          ?>
       <tr>
         <td width="174" class="first"><?php echo "键名: ".$name; ?></td>
         <td width="437" align="left"><?php
        if(is_array($value)) {                //输出元素的值
            echo "数组值为: ";print_r($value);
        }
        else{
            echo "值为: ". $value;
        }?></td>
       </tr>
       <?php } ?>
    </table>
<?php } ?>
```

在第一次运行时由于未提交，$_POST 数组为空，所以不会有任何输出。当在表单中单击“提交”按钮之后，$_POST 数组中会有一个名为 submit 的键，此时会执行 if 中的语句。

图 9-8 所示为使用上述代码遍历后 9.1.1 节表单的运行效果。

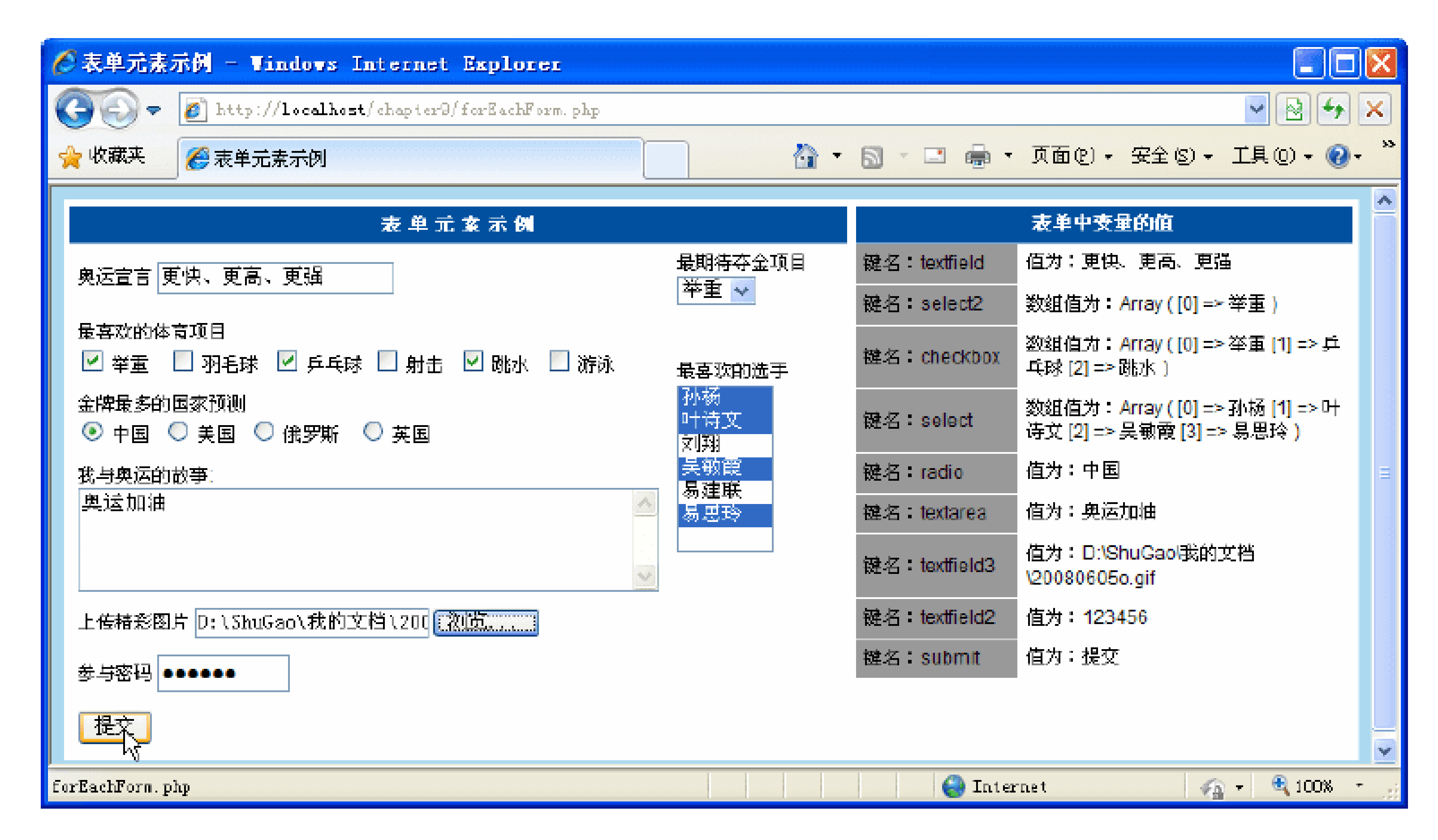


图 9-8　遍历表单变量效果

9.3.2 获取表单中的多值

从 9.3.1 节的遍历结果中可以看到，有的键对应的是单值，有的键值则是一个数组。这是因为，在表单中包含的元素类型有很多，有一些输入项需要具有一个名称（像单选按钮），而也有一些具有多个名称（像复选按钮），而其他的元素则具有唯一名称。

【实践案例 9-4】

制作一个包含单选和多选的表单，将表单使用 POST 方式进行提交。然后在页面上显示表单的提交结果，具体步骤如下。

（1）创建一个名为 case.php 的文件作为趣味测试表单的显示页面。

（2）在页面的合适位置使用表单元素制作一个趣味测试表单，本示例中使用的代码如下所示：

```
<h2>趣味测试</h2>
<form id="form1" name="form1" method="post" action="case.php">
  <p>1、如果笔相对于写字，那么书相对于 （）  。<br />
     <input type="radio" name="test1" id="radio2" value="A" />
     <label for="radio2">A 娱乐 </label>
     <input type="radio" name="test1" id="radio3" value="B" />
     <label for="radio3">B 阅读 </label>
     <input type="radio" name="test1" id="radio4" value="C" />
     <label for="radio4">C 学文化 </label>
     <input type="radio" name="test1" id="radio1" value="D" />
     <label for="radio1">D 解除疲劳</label>
  </p>
  <p>2、选择你最喜欢的神话故事 （）  。<br />
     <input name="story[]" type="checkbox" id="story" value="十二星座神话
     故事>
     <label for="story">十二星座神话故事</label>
     <input name="story[]" type="checkbox" id="checkbox" value="西王母和她
     的蟠桃仙子" />
     <label for="checkbox">西王母和她的蟠桃仙子</label>
     <input name="story[]" type="checkbox" id="checkbox2" value="普罗米修斯>
     <label for="checkbox2">普罗米修斯</label>      <br />
     <input name="story[]" type="checkbox" id="checkbox3" value="樱桃花树"
     <label for="checkbox3">樱桃花树</label>
     <input name="story[]" type="checkbox" id="checkbox4" value="森林公主"
     <label for="checkbox4">森林公主</label>
     <input name="story[]" type="checkbox" id="checkbox6" value="五路神传说>
     <label for="checkbox6"> 五路神传说</label>
     </p>
     <p>姓名:  <!-- 姓名输入域 -->
     <input name="username" type="text" id="textfield" value="您的姓名"
     size20">
```

```
        <!-- 班级选择按钮 -->
        <input type="radio" name="grade" id="radio" value="大班"> 大班&nbs 
        <input type="radio" name="grade" id="radio" value="小班">小班</p>
      </p>
    <p><input type="submit" name="submit" id="button" value="提交" /> </p>
</form>
```


如上述代码所示，该表单的 action 为 case.php，即提交到本页面进行处理。在表单中第 1 题的多个答案（单选按钮）都具有相同的名称 test1，选择班级的“大班”和“小班”也具有一个名称 grade。针对这些表单元素可以使用 name 属性从提交数组中获取，结果是一个值。

而对于第 2 题的故事名称也是多个答案（复选按钮）具有相同的名称 sotry[]。对于这种多选的情况，使用 name 属性获取之后得到的是一个数组，其中包含选择的每个值。

（3）在表单的下方编写代码，获取用户单击“提交”后的值并输出结果。这部分代码如下所示：

```
<?php
 if(isset($_POST["submit"]))                              //单击“提交”按钮
 {
    $username=$_POST['username'];                         //保存用户名
    $grade=$_POST['grade'];                               //保存班级
    $message="来自 $grade 的 <strong> $username</strong>  小朋友";
?>
  感谢您参与测试。<br /><?php echo($message."的信息如下: "); ?><br />
  第 1 题的选择是: <?php echo $_POST['test1'] ?><br />
  最喜欢的神话故事有: <br />
  <?php
     foreach($_POST['story'] as $s) {                     //遍历 story 数组
        echo($s."、 ");
     }
  } ?>
```

在测试表单中，创建了一个复选框组按钮，为了让 PHP 识别赋给一个表单变量的多个值，需要把表单中具有多个值的 name 属性命名为带有“[]”的名称。例如，代码中为复选框的命名为“story[]”。这样 PHP 将像处理所有其他数组一样对待所提交的变量。

（4）在浏览器中运行 case.php 文件。然后对表单进行操作，最后单击“提交”按钮查看结果，如图 9-9 所示。

9.3.3 动态生成表单

前面创建的表单都是静态的，而在一些网站中需要使用 PHP 根据不同的请求从数据库读取数据来动态生成表单，像随机出题。本节将介绍几种动态生成表单元素的方法。

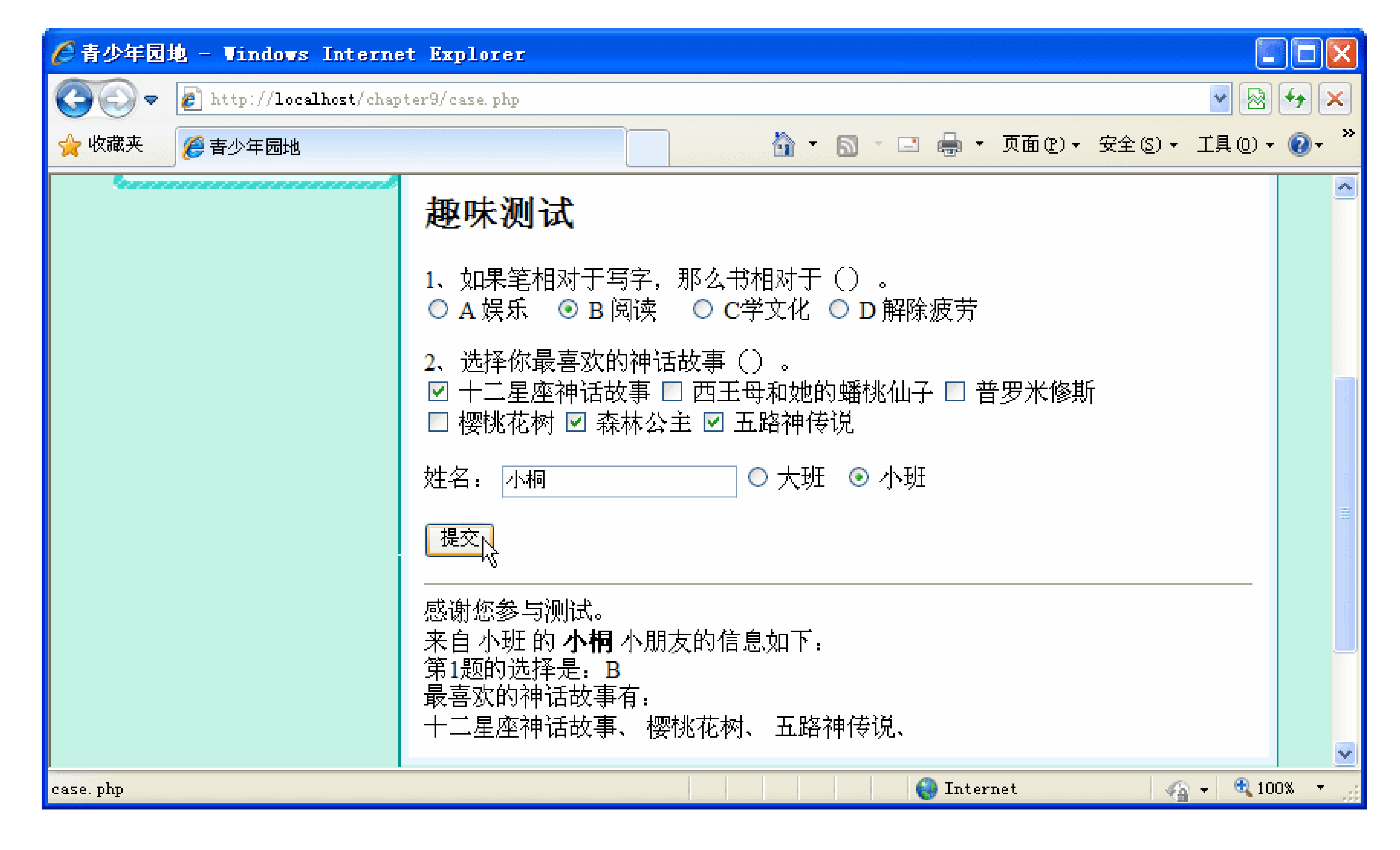


图 9-9　测试表单运行效果

1. 生成一组单选按钮

单选按钮使用户可以在多个选项中选择一项，类似于单项选择题。一个单选按钮的 HTML 代码如下：

```
<input type="radio" name="name" id="id" value="value" />
```

Input 标记的 type 属性设置为 radio 表示一个单选按钮。相同 name 的单选按钮表示一个单选按钮组，其中只能有一个被选中。id 属性用于唯一标识一个单选按钮，value 属性用于指定选中该按钮之后的取值。

例如，下面的代码创建了 4 个单选按钮，它们使用相同的组名 season。

```
<input type="radio" name="season" id="spring" value="春" />春
<input type="radio" name="season" id="summer" value="夏" />夏
<input type="radio" name="season" id="autumn" value="秋" />秋
<input type="radio" name="season" id="winter" value="冬" />冬
```

对于单选按钮来说最重要的是 name 属性和 value 属性。为了可以动态地生成它们，这里定义了一个函数 generateRaidoButtons()，实现代码如下：

```
//动态生成一组单选按钮函数
function generateRaidoButtons($name,$data=array(),$default=""){
  $name=htmlentities($name);                        //转换名称
  $html="";                                    //准备字符串
  foreach ($data as $value=>$label) {                      //开始遍历
        $value=htmlentities($value);                     //转换名称
        $html.="<input type=\"radio\"";           //生成单选按钮的开始
        if($value==$default) $html.=" checked ";    //判断是否选中
        $html.="name=\"$name\" value=\"$value\">";  //生成名称和值
        $html.=$label."<br/>";                      //显示文本
```

```
        }
        return $html;
    }
```

如上述代码所示，生成函数有 3 个参数：$name 参数用于指定生成单选按钮的名称，$data 参数接收一个关联数组（默认为空数组），$default 参数用来指定哪个单选按钮被选中（默认为空）。在函数体内首先把传递的$name 进行 HTML 实体转换，然后使用 foreach 语句对$data 数组进行遍历，在遍历时根据元素的名称和值生成单选按钮的 HTML 代码，如果某个值与$default 参数相同，则增加一个 checked 输出表示默认选中。

下面的代码就调用 generateRaidoButtons()函数生成一个包括 4 个单选按钮的 HTML。

```
<p>1、找出不同类的一项 （）  。<br />
<?php
 //定义一个要作为单选按钮的数组
  $question=array("A"=>"碟子","B"=>"米饭","C"=>"小勺","D"=>"铁锅");
  echo generateRaidoButtons("ques",$question,"A");      //显示单选按钮
?> </p>
```

上述代码指定单选按钮的组名为 ques，使用的数据来自$question 数组，第三个参数表示默认选择 A。生成的 HTML 代码如下所示：

```
<input type="radio" checked name="ques" value="A">碟子<br/>
<input type="radio"name="ques" value="B">米饭<br/>
<input type="radio"name="ques" value="C">小勺<br/>
<input type="radio"name="ques" value="D">铁锅<br/>
```

图 9-10 所示为最终生成单选按钮的运行效果。

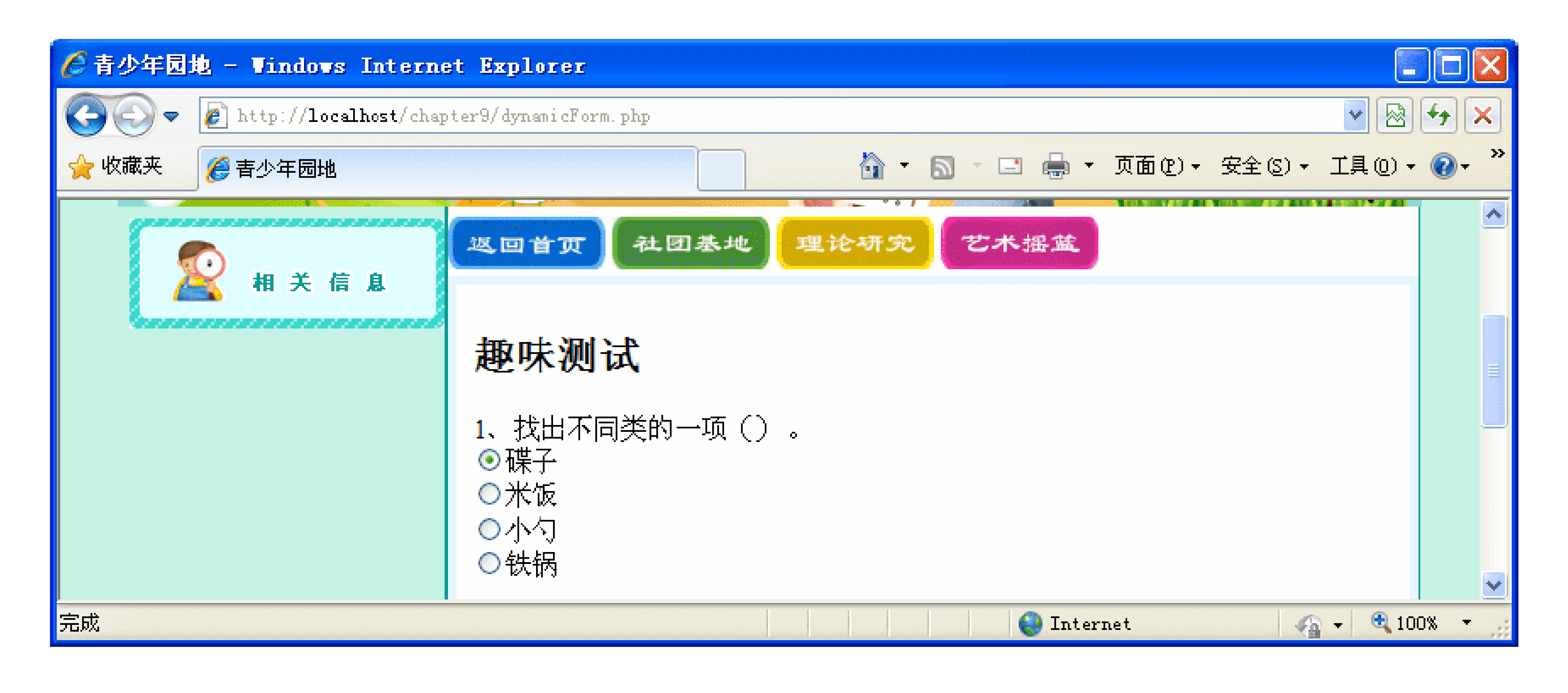


图 9-10　生成单选按钮效果

2．生成一组复选按钮

复选按钮在表单中适用于可以选择多项的情况。一个复选按钮的 HTML 代码如下：

```
<input type="checkbox" name="checkbox" id="checkbox">
```

Input 标记的 type 属性设置为 checkbox 表示是一个复选按钮。相同 name 的复选按钮表示一组，其中至少能有一个被选中。id 属性和 value 属性与其他标记相同。

例如，下面的代码创建了 4 个复选按钮，它们使用相同的组名 ball。

```
属于三小球的有 ()。
   <input type="checkbox" name="ball" value="A" />乒乓球, ,
   <input type="checkbox" name="ball" value="B" />网球
   <input type="checkbox" name="ball" value="C" />羽毛球
   <input type="checkbox" name="ball" value="D" />高尔夫球
```

根据上面的代码编写一个函数 generateCheckButtons()动态生成一组复选按钮，实现代码如下：

```
 //动态生成一组复选按钮函数
 function generateCheckButtons($name,$data,$default=NULL)
{
    $html="";
    if(!is_array($default)) $default=array();
    foreach ($data as $value=>$label) {
        $value=htmlentities($value);
        $html.="<input type=\"checkbox\"";
        if(in_array($value,$default)) $html.=" checked ";
        $html.=" name=\"$name\" value=\"$value\">";
        $html.=$label."<br/>";
    }
    return $html;
}
```

generateCheckButtons()函数接收 3 个参数，分别是复选框名称$name 参数，数据来源数组$data 参数，默认选项数组$default 参数（可省略）。在函数内，首先判断$default 参数是不是数组，如果不是则赋一个空的数组表示没有默认选中项。接下来对$data 数组进行遍历，每遍历一次都会生成复选按钮的 HTML 代码，同时判断如果当前值在$default 参数中，则增加 checked 输出表示选中。

例如，下面的代码演示如何调用 generateCheckButtons()函数生成一个包括 8 个复选按钮的 HTML。

```
<p>2、中国的四大名著分别是 () 。<br />
    <?php
        //定义数组，其中键名为复选按钮的值，键值为复选按钮的文本
        $books=array("ym"=>"一帘幽梦","xyj"=>"西游记","tlbb"=>"天龙八部",
"hlm"=>"红楼梦","yttl"=>"倚天屠龙记","xhz"=>"水浒传","xqj"=>"寻秦记
","sgyy"=>"三国演义");
        //定义一个要默认选中的数组，这里指定的是数组的键名
         $selected=array("xyj","sgyy");
        //动态生成并输出
```

```
        echo generateCheckButtons("books",$books,$selected);
    ?> </p>
```

运行之后将看到如图 9-11 所示的效果，其中默认有两个选项值。通过查看源代码可以看到上面函数生成的 HTML 代码，如图 9-12 所示。

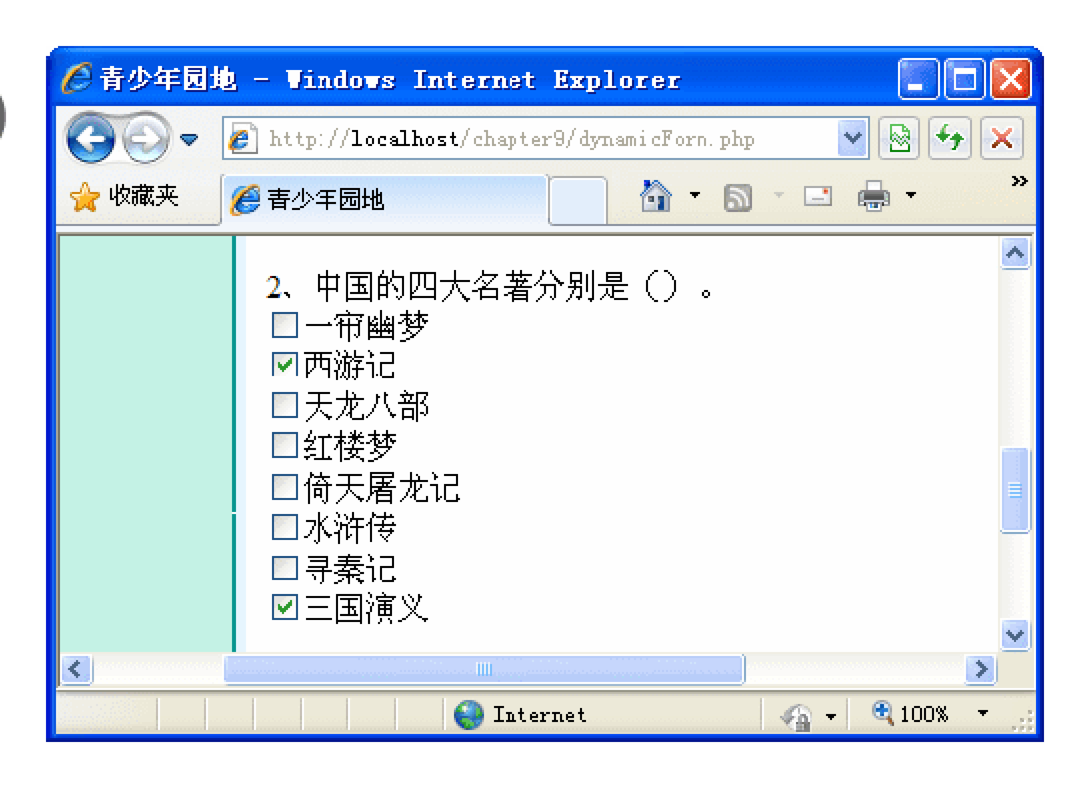


图 9-11　生成复选按钮效果



图 9-12　查看生成的复选按钮代码

3．生成一组下拉列表

下拉列表也称为下拉菜单，使用 select 标记进行定义，在标记中嵌套 option 标记来表示一个列表（菜单）项，其中 value 属性表示选中的值。使用 selected 标识可以指定默认选中的项。

下面的一个下拉列表的示例代码。

```
<select name="select" id="select">
  <option value="0371">郑州</option>
  <option value="010" selected>北京</option>
  <option value="0731">长沙</option>
</select>
```

上述代码运行后会默认选中“北京”。select 标记有一个 multiple 属性，设置为 multiple 时表示可以在列表中多选。

综合前面的例子，只需要对函数进行部分 HTML 代码的调整就可以实现自动生成一组下拉菜单的功能。generateSelecteMenus()函数代码如下所示：

```
function generateSelecteMenus($name,$data,$default=NULL)
{
    if(!is_array($default)) $default=array();
    $html="<select name=\"$name\" id='".$name."[]' multe=\"multiple\">";
    foreach ($data as $value=>$label) {
        $value=htmlentities($value);
        $html.="<option value=\"$value\"";          //输出菜单项
        if(in_array($value,$default)) $html.=" selected ";
                                                     //判断是否被选中
            $html.=" name=\"$name\" value=\"$value\">";
```

```
            $html.=$label."</option><br/>";
        }
        return $html."</select>";
    }
```

可以看到，该函数同样需要 3 个参数且实现原理也相同，就不再重复。下面是调用代码：

```
    $selected=array("xyj","tlbb");                              //定义默认选中项
    echo generateSelecteMenus("select",$books,$selected);  //输出
```

运行之后会看到一个下拉列表，并且 value 是 xyj 和 tlbb 的项被选中，如图 9-13 所示。如果将 generateSelecteMenus()函数中的“multiple=\"multiple\"”去掉，将会生成下拉菜单，运行效果如图 9-14 所示。

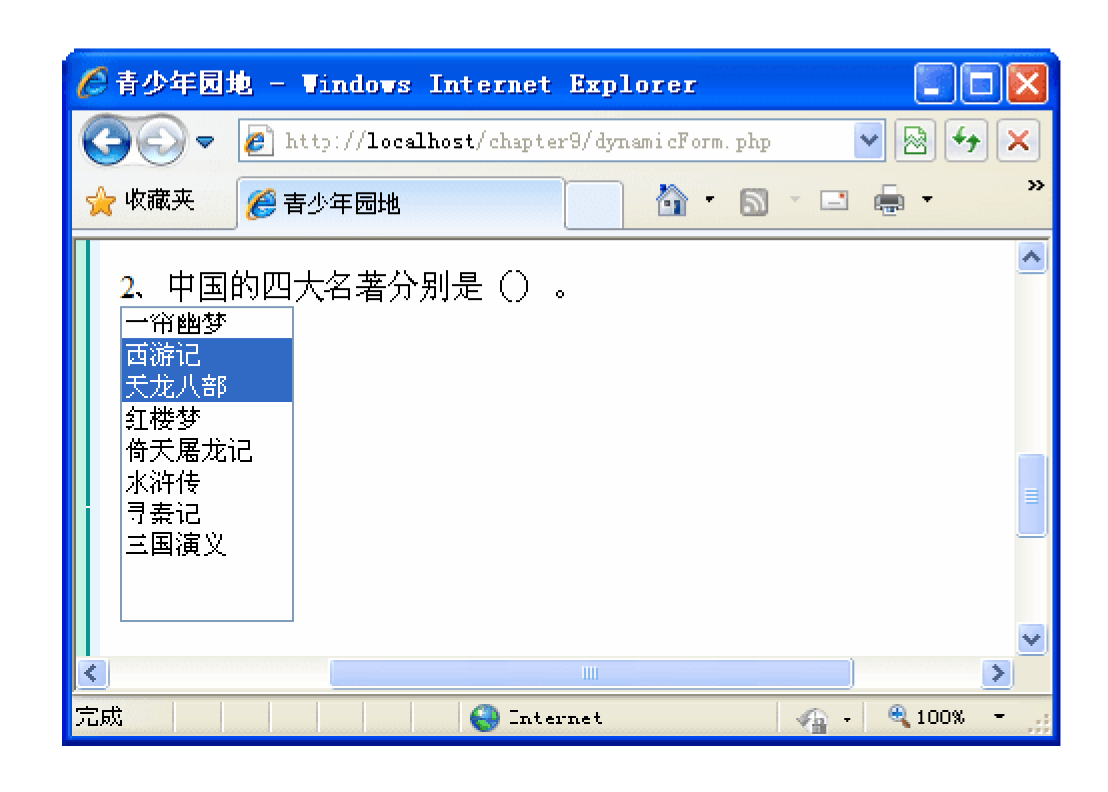


图 9-13　生成下拉列表

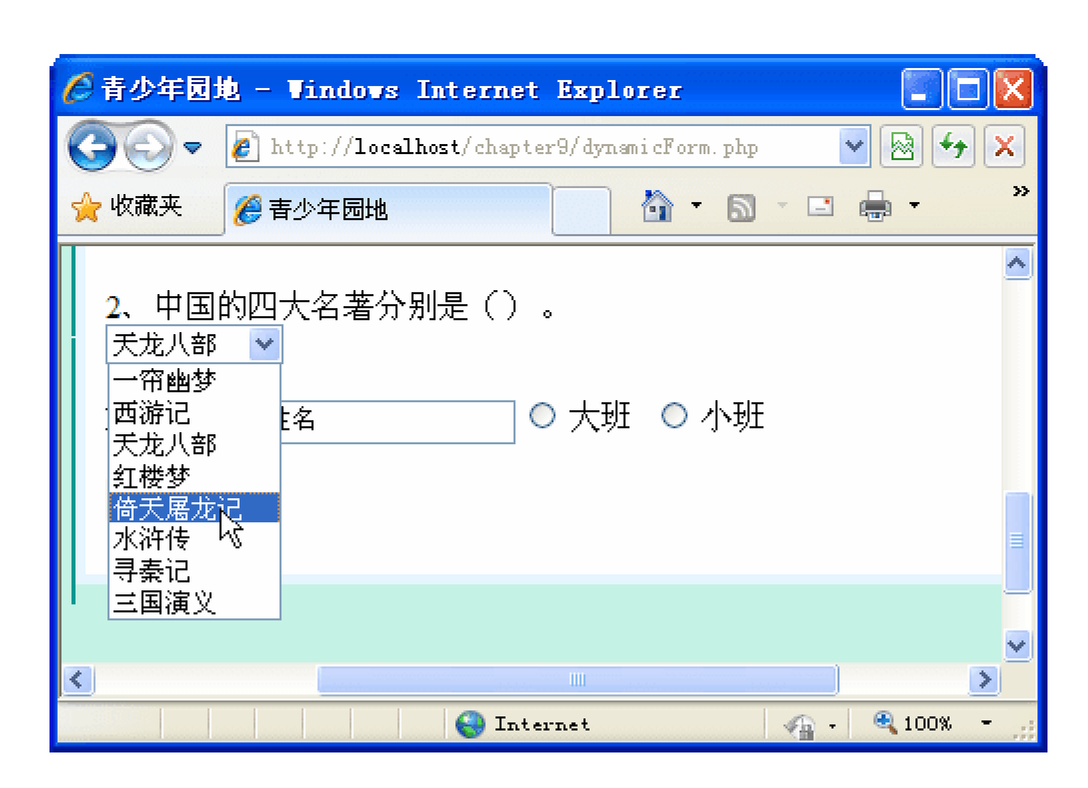


图 9-14　生成下拉菜单

9.4　表单处理技巧

出于安全的考虑，经常需要对表单进行一些特殊的处理。例如，对于填写敏感数据的表单判断是否从本站进行的提交、判断一个投票表单是否已经提交过了或者过期了。本节将介绍这些情况下的表单处理技巧。

9.4.1　检测表单提交路径

通过对表单提交来源的处理可以只允许从某个域名（URL 地址）或脚本自身进行提交，从而可以有效防止伪造相同的表单向程序进行提交造成的安全问题。

在 PHP 中提供了一个名为$_SERVER 的全局数组，其中的 HTTP_REFERER 键保存的是上一页的来源，例如表单的提交地址或者是转向前的 URL 地址。如果用户从其他的计算机上提交表单或者从浏览器中直接输入表单提交地址，该键会是表单的来源或者空值，这样就可以通过该值进行处理。

如下列出了$_SERVER 数组中常用的键及含义。

- **$_SERVER['HTTP_REFERER']　值是一个完整的 URL 地址，用于标识前一个页面的来源。**
- **$_SERVER['SERVER_NAME']　值是当前的域名（服务器名称）。**
- **$_SERVER['PHP_SELF']　值是当前 URL 中除去域名的完整路径，包含文件名。**
- **$_SERVER['REQUEST_METHOD']　访问页面时的请求方法。如 GET、HEAD、POST 或者 PUT。**

例如，对于“http://localhost/chapter9/index.php”地址使用$_SERVER['SERVER_NAME']返回 localhost，使用$_SERVER['PHP_SELF']返回/chapter9/index.php。

假设，在 case.php 文件中有如下的表单代码：

```
<form id="form1" name="form1" method="post" action="case.php">
    <input type="submit" name="submit" id="button" value="提交" />
</form>
```

上述代码指定表单以 POST 方式提交到本页面，即由 case.php 进行处理。为了判断用户是否正常提交，可以使用如下代码：

```
<?php
$pathname=$_SERVER['PHP_SELF'];                     //保存当前页面名称
if($_SERVER['REQUEST_METHOD']=='POST')              //是否为 POST 提交
{
    $ref=$_SERVER['HTTP_REFERER'];                  //获取表单提交前的 URL
    $url="http://{$_SERVER['SERVER_NAME']}$pathname";  //获取当前的 URL
    echo "当前页面来源为".$ref."<br/>服务器地址为: ".$url;
    if(strcmp($url, $ref)==0){                      //比较两个 URL 是否相同
        echo "<br/>正常提交";
    }else    {
        echo "<br/>不允许外站提交的数据";
    }
} else{
    echo "<br/>请先提交表单再操作。";
}
?>
```

上述代码通过比较 http://{$_SERVER['SERVER_NAME']}$pathname 是否与$_SERVER['HTTP_REFERER']相同来判断是否为合法的表单提交动作。

9.4.2　避免表单重复提交

在用户提交表单时由于网速的原因，或者网页被恶意刷新，可能会导致同一个表单重复提交，造成数据重复，这个是比较棘手的问题。对程序开发人员来说可以从客户端和服务器端两个方面来避免。

1. 使用客户端脚本

客户端脚本通常用于对表单的数据进行有效性验证，同样也可以使用它处理表单的重复提交问题。例如，下面的示例代码：

```
  <form id="form1" name="search" method="post" action="search.php">
    关键字: <input name="key" type="text" id="key"  size="20"/>
      <input type="submit" name="submit" id="button" value="搜索"
         onclick="document.search.submit.value='正在提交';
                 document.search.submit.disabled=true; document.search.
                 submit();"   />
</form>
```

上述代码为“搜索”按钮添加了 onclick 事件检测用户的单击状态。一旦用户单击“搜索”按钮之后该按钮会变得不可用，并显示“正在提交”提示。用户不能再次单击该按钮也就避免了表单的重复提交。

第二种实现方式是利用表单的 onSubmit 事件实现，代码如下所示：

```
<script language="javascript" type="text/javascript">
var isFirst=0;                      //第 1 次提示标识
function checkSubmit(form)
{
    if(isFirst==0) {                //如果是第 1 次
        isFirst++;                  //修改标识的值
        return true;                //返回 true，可以提交
    }else{
        alert("表单已经提交，请不要重复操作.");
        return false;               //返回 false，表单不会提交
    }
}
</script>
<form id="form1" name="search" method="post" action="search.php"
onsubmit="return checkSubmit(this)">
  关键字: <input name="key" type="text" id="key"  size="20"/>
  <input type="submit" name="submit" id="button" value="搜索" />
</form>
```

在上述代码中如果用户第一次单击“搜索”按钮，onsubmit 事件指定的脚本会自动调用，并在脚本中修改 isFirst 变量的值，然后提交表单。当再次单击“搜索”按钮时，由于 isFirst 变量的值已经不是 0，会提示已经提交，并取消提交动作。

2. 使用 Cookie 处理

这种方式是指在表单提交之后使用 Cookie 记录已提交，实现代码如下所示：

```
<?php
```

```
if(!isset($_POST['submit'])){                    //如果单击了“提交”按钮
    setcookie("tempcookie","",time()+30);         //记录提交状态
    header("Location:".$_SERVER[PHP_SELF]);       //转向提交页面
    exit();
}else{
    setcookie("tempcookie","",0);
    echo "表单已经提交，请不要重复操作";
}
?>
```



这种方式有一个明显的缺陷就是如果客户端禁用了 Cookie，代码将不起任何作用。

3．使用 Session 处理

这是 PHP 服务端的解决方案。Session 保存在服务器端，可以在 PHP 运行过程中修改 Session 的值，待下次访问这个变量时，得到的是新赋的值。所以，可以用一个 Session 变量保存表单提交的值，如果不匹配则认为在重复提交。具体实现代码如下：

```
<?php
session_start();                    //开始一个会话
$code=mt_rand(0,100);               //产生一个随机数
$_SEEION['code']=$code;             //保存这个随机数
?>
```

上面代码会在表单页面产生一个随机数，将它作为隐藏域放在表单内，代码如下：

```
<input name="checkcode" type="hidden" value="<?php echo $code;?>"/>
```

然后在表单提交后的处理页面中获取这个随机数，如果相同则是正常提交，否则提示不能提交，代码如下：

```
<?php
session_start();                                  //开始一个会话
if($_POST['checkcode']==$_SESSION['code']){//判断提交表单的随机数是否相同
    unset($_SESSION['code']);                     //清空会话
    echo "表单提交正常";
}
else{
    echo "表单已经提交，请不要重复操作";
}
?>
```

4．使用 header 函数

除了上面的方法之外，还有一个更简单的方法，就是当用户提交表单服务器端处理之

后立即转向其他页面。示例代码如下：

```
<?php
if(isset($_POST['action'])&&$_POST['action']=='submit'){
    header('Location:submit_succes.php');
}?>
```

这样即使用户刷新页面，也不会导致重复提交，因为已经转向新的页面，而这个脚本也已经不会提交数据了。

9.4.3　表单过期处理

在开发过程中经常会出现因表单出错而重新返回页面的情况，此时前面填写的信息全部会丢失。为了使数据支持页面的后退，可以通过如下两种方式实现。

- **使用 header 头设置缓存控制头 Cache-Control。**

```
header('Cache-Control:private, must-revalidate');    //支持页面后退
```

- **使用 session_cache_limiter 方法。**

```
session_cache_limiter('private,must-revalidate');
                                        //此方法写在 session_start 方法之前
```

下面的代码可以防止用户填写表单后，单击“提交”按钮返回时刚刚在表单上填写的内容被清除。

```
session_cache_limiter('nocache');
session_cache_limiter('private');
session_cache_limiter('public');
session_start();
```

将上述代码放在所有脚本之前，这样用户在返回该表单时已经填写的内容就不会被清空。

下面简单介绍 Cache-Control 消息头的作用，它用于指定请求和响应遵循的缓存机制，而且在请求或者响应信息中设置 Cache-Control 并不会修改另一个消息处理过程中的缓存过程。表 9-3 中列出了该消息常用的指令及其作用。

表 9-3　Cache-Control 消息常用的指令

指令名称	说明
Public	指定响应可被任何缓存区缓存
private	指定对于单个用户的整个或者部分响应信息，不能被共享缓存处理。这允许服务器仅描述当前用户的部分响应信息，此响应信息对其他用户的请求无效
no-cache	指定请求或者响应信息不能缓存
no-store	用于防止重要的信息被无意地发布。在请求信息中发送将使得请求和响应信息都不使用缓存
max-age	指定客户端可以接收生存期不大于指定的响应

续表

指令名称	说明
min-fresh	指定客户端可以接收响应时间小于当前时间加上指定时间的响应
max-stale	指定客户端可以接收超出超时间期的响应信息。如果指定 max-stale 消息的信息的值，那么客户端可以接收超出超时期指定值的响应信息

9.5 转换 URL 中的汉字

使用 GET 方式提交表单时，数据的值必须为 ASCII 字符。因此，当 GET 方式传递的字符串含有汉字或者其他非 ASCII 字符时，则需要使用额外的转换操作。这包括传递时对汉字进行编码，获取时对汉字进行解码。PHP 提供了 urlencode()和 urldecode()两个函数实现这个过程。

9.5.1 编码操作

编码主要是指对地址栏传递参数进行的一种编码规则。例如，在参数中带有空格传递时就会发生错误，而用 URL 编码过以后，空格转换成了"%20"，这样错误就不会发生。对中文进行编码也是同样的情况。

PHP 中对 URL 进行编码使用的是 urlencode()函数，该函数的语法如下：

```
string urlencode ( string $str )
```

该函数可以实现将字符串$str 进行 URL 编码并返回编码后的字符串，它与 POST 方式提交数据采用的编码方式是一样的。

【实践案例 9-5】

本实例中单击“搜索”链接将通过 URL 传递中文关键字到指定页面，在传递之前使用 urlencode()函数对关键字进行 URL 编码。这样一来显示在 IE 地址栏中的字符串是 URL 编码后的字符串。

实现代码如下所示：

```
<h2>站内搜索</h2>
 当前正在搜索“儿童读物”的内容。您也可以通过如下链接直接转到结果列表：<br/>
 <a href="http://www.itzcn.com/link.php?name=<?php echo urlencode("儿童
 读物");?>">搜索儿童读物</a><br/>   http://www.itzcn.com/link.php?name=
 <?php echo urlencode("儿童读物");?>
```

上述代码中，使用 urlencode()函数对关键字“儿童读物”进行编码。另外，下面还显示了编码后的 URL 字符串，运行效果如图 9-15 所示。



图 9-15　对 URL 中的汉字编码



对于服务器而言，编码前后的字符串并没有什么区别，服务器能够自动识别。

9.5.2　解码操作

使用$_GET 可以获取 URL 中传递的参数，但是对于进行编码后的 URL 查询字符串，则需要通过 urldecode()函数对获取后的字符串进行解码。

该函数的语法如下：

```
string urldecode( string $str)
```

该函数可以实现将 URL 编码$str 查询字符串进行解码。

【实践案例 9-6】

在 9.5.1 节使用 urlencode()函数实现了对“儿童读物”字符串进行编码，将编码后的字符串传给变量 name。在本实例中将应用 urldecode()函数对获取的变量 name 进行解码，将解码后的结果输出到浏览器。

实现代码如下所示：

```
<h2>站内搜索</h2>
<?php
if(isset($_GET['name'])){
  $name=urldecode($_GET['name']);
  }?>
您输入的关键字是“<?php echo $name;?>”。<br/> 您可以通过如下链接搜索<?php echo
$name;?>:
<br/>   <a href="http://www.itzcn.com/link.php?name=<?php echo
$name;?>">
          http://www.itzcn.com/link.php?name=<?php echo $_GET['name'];?>
          </a>
```

将代码保存为 urldecode.php，然后再使用如下 URL 传递 name 参数，运行后将看到如图 9-16 所示的效果。



图 9-16　对 URL 中的汉字解码

```
urldecode.php?name=%E5%84%BF%E7%AB%A5%E8%AF%BB%E7%89%A9
```

9.6　Cookie 存储数据

由于 HTTP 本身是一个无状态的连接协议，因此当用户访问一个网站时，也就无法保存和记录每个用户的状态。为了支持客户端与服务器端的这种交互，就需要通过某种技术为用户和网站之间的交互保存状态。

使用前面介绍的表单仅仅能实现客户端向服务器的数据提交。为此 PHP 提供了 Cookie 和 Session 两种解决方案，它们既有区别又有相同之处，主要功能都是把客户端与服务器关联起来。本节首先介绍 Cookie 的使用。

9.6.1　Cookie 概述

Cookie 最根本的作用是帮助站点开发人员保存有关访问者的信息，或者说 Cookie 是一种保持 Web 应用程序连续性（即执行“会话管理”）的方法。因为浏览器和服务器除了在短暂的信息交换阶段以外总是断开的，而用户向服务器发送的每个请求都是单独处理的，与其他所有请求无关。然而在大多数情况下，都有必要让服务器在用户请求某个页面时对用户进行识别。

当用户浏览某网站时，网站存储在用户机器上的一个小文本文件记录了用户 ID，密码、浏览过的网页、停留的时间等信息。当再次访问该网站时，网站通过读取 Cookie，获取用户的相关信息，就可以做出相应的动作，如在页面显示欢迎标语，或者让用户不用输入 ID、密码就直接登录等。

如下是使用 Cookie 的一些常见应用。

- **网站能够精确地知道有多少人浏览过。**

- **网站保存用户的设置，按照用户的喜好定制网页外观。**
- **电子商务站能够实现像“购物篮”、“快速结账”这样的功能。**

Cookie 随网页 HTTP 的 Header 信息进行传递，浏览器的每一次网页请求都可以伴随 Cookie 传递。例如：浏览器的打开或网页刷新等操作。服务器将 Cookie 添加到网页 HTTP 的 Header 信息中，随网页数据传到浏览器中，浏览器会根据电脑中的 Cookie 设置选择是否保存这些数据。如果浏览器不允许 Cookie 保存，浏览器关闭之后，这些数据就会消失。

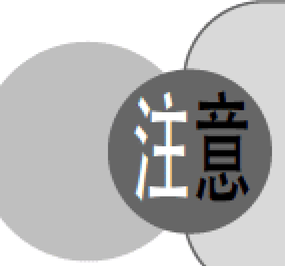
Cookie 有如下两种类型。

- **临时 Cookie　此 Cookie 只是在浏览器上保存一段规定的时间，一旦超过这个规定的时间，该 Cookie 就会被系统清除。**
- **持续 Cookie　此 Cookie 保存在用户的 Cookie 文件中，等到下一次用户访问时依然可以调用。**

那么如何为 Cookie 设置有效时间呢？该有效期应为多长时间？这些都可以通过 Cookie 属性来设置，表 9-4 列出了这些属性。

表 9-4　Cookie 属性

属性名称	说明
name	指定 Cookie 的名称，必须
value	指定该 name 对应的值，保存的形式为 name=value
expires	指定 Cookie 的过期日期和时间，保存的形式为 expires=日期，必须
path	指定 Cookie 在哪些路径下有效，默认为“/”，即 domain 属性指定域名下所有页面有效
domain	指定 Cookie 作用于的域名
secure	一个布尔值，如果为 true 表示只在 SSL 加密连接时才发送 Cookie 到客户端



由于 Cookie 是保存在客户端的文件中，因此不适合保存敏感的数据。另外，在编写程序时应该避免过度依赖 Cookie，如果不能获取 Cookie 的值应该有其他实现方案。

9.6.2　向 Cookie 保存数据

PHP 提供了 setcookie()函数，可以非常简单地向 Cookie 中保存数据。该函数的语法格式如下所示：

```
bool setcookie (string $name [, string $value [, int $expire [, string $path
[, string $domain [, int $secure]]]]])
```

由语法格式得知，该函数的所有参数中除了$name 参数外都是可选的。如果只有一个$name 参数，则表示删除客户端浏览器中名字为$name 的 Cookie。也可以通过设置为空字符串来跳过一个参数，但$expire 和$secure 的值必须为整数，不能用空字符串。$expire 参数是一个由 time()或者 mktime()函数返回的 UNIX 格式时间。$secure 表示 Cookie 只在建立 HTTP 连接时才发送。



使用 time()函数可以返回从 1970 年 1 月 1 日以来按秒计算的经过时间。可以添加一个偏移值，它指定可以访问 Cookie 的时间。

例如，要创建一个名为 userskin 的 Cookie，其中保存的值是 winxp，并且这个 Cookie 在创建后的一小时（3600 秒）内是可用的。具体语句如下：

```
setcookie("userskin", "winxp", time()+3600);
```

除了使用 time()函数外还可以使用 mktime()函数指定过期时间。该函数的具体格式如下所示：

```
int mktime ( [int $hour [, int $minute [, int $second [, int $month [, int
$day [, int $year [, int $is_dst]]]]]]])
```

使用上述格式根据给出的参数返回 UNIX 时间戳。其中，参数可以从右向左省略，任何省略的参数会被设置成本地日期和时间的当前值。

例如，下面的语句将创建一个 Cookie，它将在 2012 年 10 月 11 号的第一秒钟过期：

```
setcookie("lastlogintime", "2012- 10-10", mktime(0, 0, 1, 10, 11, 2012));
```

【实践案例 9-7】

根据上面介绍的 Cookie 创建语法，创建一些 Cookie 的示例。具体代码如下所示：

```
//简单创建一个 cookie
setcookie("title"," 4 大主题夏令营:英语 篮球 减肥 拓展热报");
//创建一个带有有效时间为两个小时 (7200 秒) 的 cookie
setcookie("category","夏令营 | 中小学",time()+7200);
//创建一个完整的 cookie
setcookie("more","http://www.itzcn.com/social/more",time()+7200,"/socia
l",".itzcn.com",1);
```

当然，也可以通过在 Cookie 名称中使用数组符号来保存数据。这样便可以在数组中添加多个元素到 Cookie 中，然后一次性获取并处理。这种方式的示例如下所示：

```
//在 Cookie 中以数组形式保存数据
setcookie("videos[0]","2013 海文考研英语基础班视频");
setcookie("videos[1]","2013 海文考研政治基础班视频");
setcookie("videos[2]", "青青部落夏令营宣传片");
setcookie("videos[3]", "澳美加国际冬令营精彩视频");
```

【实践案例 9-8】

Cookie 创建完成之后不会立即生效，而是等到请求下一个页面时才能生效。这是因为设置好的 Cookie 会由服务器端传递给客户端，在请求下一个页面时，客户端才能将 Cookie 取出并传回服务器端。

下面代码使用 Cookie 实现了判断用户是否第一次访问网站的功能。

```
<?php
if(isset($_COOKIE['visited'])){
    setcookie("visited","1",mktime()+86400,"/") or die("您的浏览器不支持或者
    禁用了 Cookie。");
    echo "您好，第一次访问本站。";
} else{
    echo "感谢您再次光临本站。";
}
?>
```

运行上述代码时，如果是第一次访问该页面，因为 Cookie 没有保存，所以将输出“您好，第一次访问本站”。当第二次刷新时，Cookie 会发送到服务器端，PHP 接收到客户端的 Cookie 得知不是第一次访问，此时输出“感谢您再次光临本站”。

9.6.3 从 Cookie 读取数据

如果用户已经创建了一个 Cookie，它的值会自动随 PHP 发送到客户端浏览器，并且转换为$_COOKIE 数组的键值。与前面$_GET 和$_POST 相同，都可以通过键名来获取相应的数据。

【实践案例 9-9】

前面创建了很多的示例 Cookie，现在将创建一个页面并读取 Cookie 中保存的数据。首先新建一个 PHP 页面，添加实践案例 9-7 中创建的 Cookie 代码。

在页面需要显示 Cookie 的位置添加如下代码：

```
<h2>今日头条</h2>
<ul>
    <li>标题：<?php echo $_COOKIE["title"]; ?></li>
    <li>所属分类：<?php echo $_COOKIE["category"]; ?></li>
</ul>     <hr/>
<h2>精彩视频</h2>
<?php
echo "<ul>";
foreach ($_COOKIE["videos"] as $video) {
    echo "<li>$video </li>";
}
echo "</ul>";
?>
```

在浏览器中运行 PHP 文件，查看显示 Cookie 数据的运行效果，如图 9-17 所示。

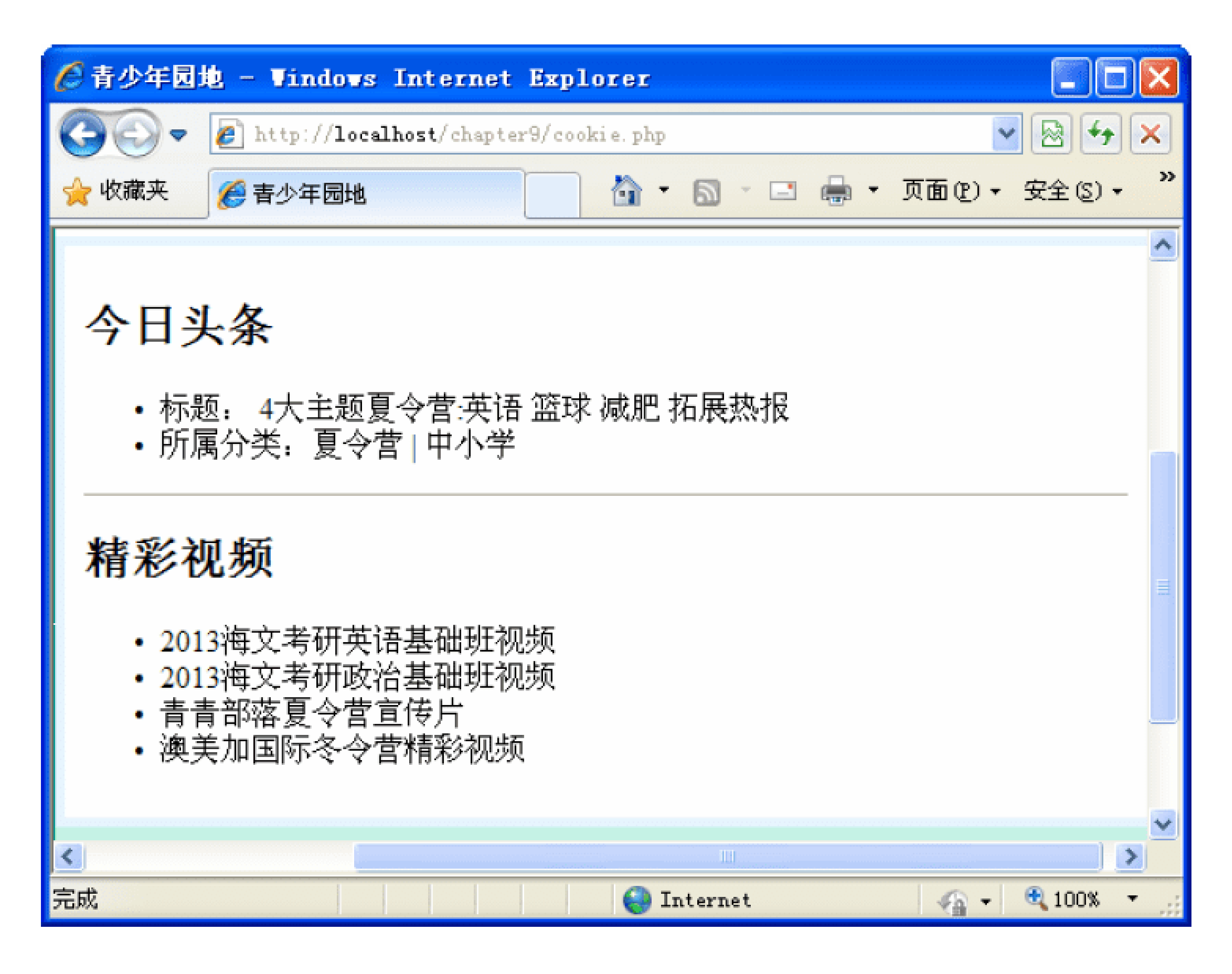
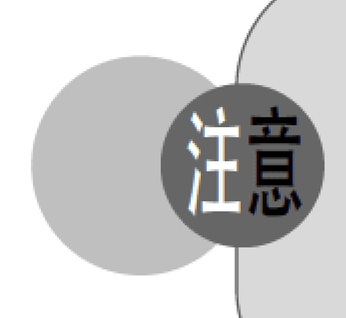


图 9-17　显示 Cookie 数据运行效果



Cookie 值是由浏览器作为 HTTP 头的一部分发送的。因此，在向浏览器发送任何输出之前，必须设置 Cookie 值。即使发送一个单独的空格也将无法设置 Cookie 值。为了避免出现问题，一定要将设置 Cookie 值的 PHP 脚本放在文件的顶部，并且保证前面没有空白字符。此外，还应该在向浏览器发送输出的 echo 语句或另一个 PHP 语句执行之前设置 Cookie 值。

9.6.4　删除 Cookie 数据

虽然 Cookie 有一个过期时间，在创建它之后的某个时刻，它将被自动删除。不过，用户也可以立即删除一个 Cookie。要想实现这一点，将 Cookie 的过期时间设置为过去的一个时间即可。

例如，要删除前面创建的名为 title 的 Cookie，可以使用如下代码：

```
setcookie("title", "", time()-3600);
```

上述语句将 Cookie 的过期时间设置为一小时（3600 秒）之前。注意，这个 Cookie 的值被指定为一个空字符串。由于这个 Cookie 不再可用，因此它的值也就不重要了。

另外，还有一个比较简单的方法，具体形式如下：

```
setcookie("title ");
```

【实践案例 9-10】

编写一个案例测试使用 setcookie()函数删除 Cookie 的效果。

同样是首先创建一个 PHP 文件，添加实践案例 9-7 的代码创建 Cookie。然后使用

setcookie()函数进行删除，具体实现语句如下所示：

```
<?php
    echo "删除键为 title 的 Cookie。<br/>";
    setcookie("title");
    echo "删除键为 videos[0]的 Cookie。<br/>";
    setcookie("videos[0]");
    echo "删除键为 videos[2]的 Cookie。<br/>";
    setcookie("videos[2]");
?>
删除后当前的 Cookie 内容如下:
<pre>
<?php print_r($_COOKIE);?>
</pre>
```

在浏览器中运行 PHP 文件，查看删除 Cookie 数据的运行效果，如图 9-18 所示。

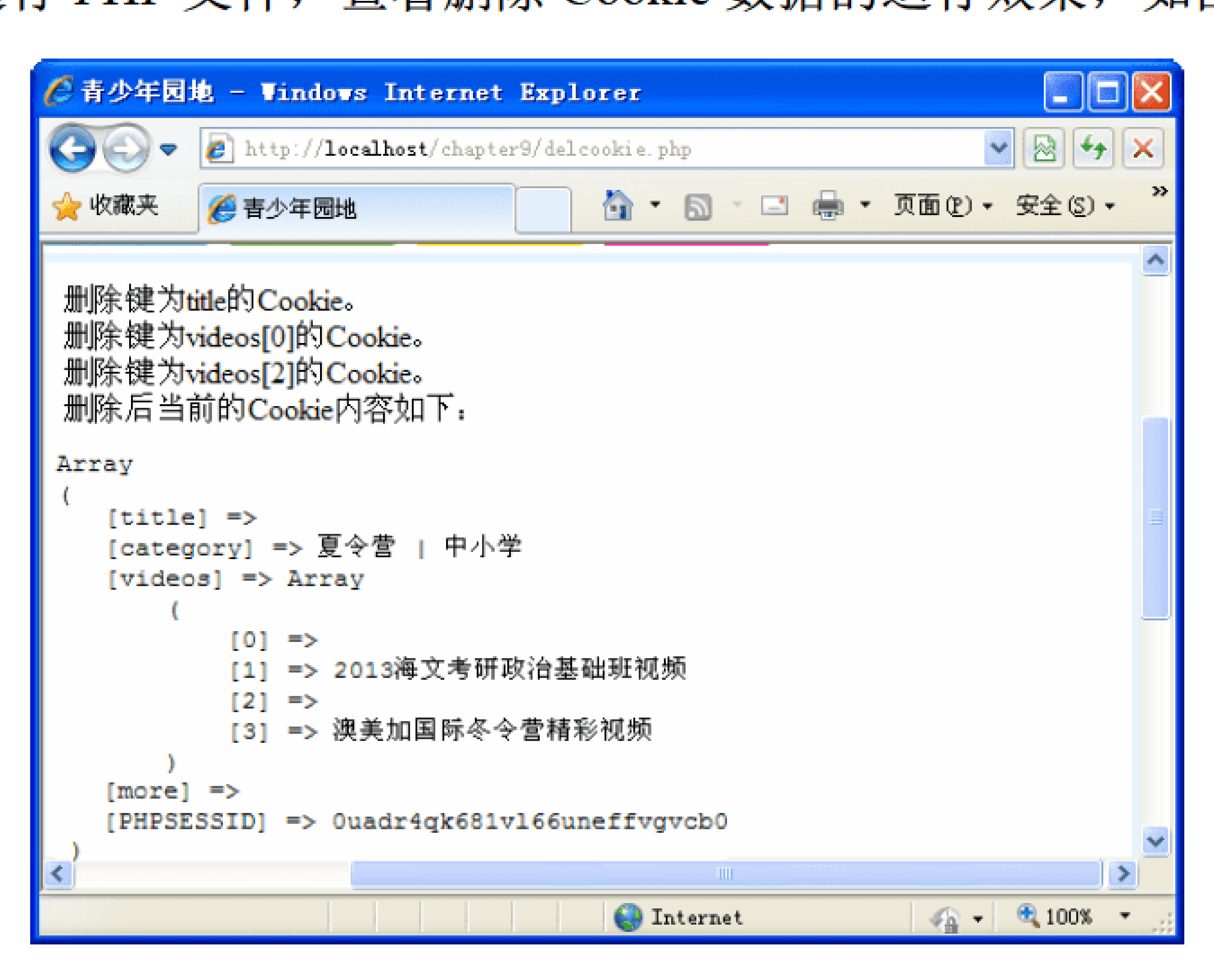


图 9-18　删除 Cookie 数据运行效果

9.7　Session 存储数据

Session 又称为会话，是指一个客户端与服务器交互进行通信的时间间隔，通常指从进入系统到退出系统之间所经过的时间。具体到 Web 中的 Session 就是指用户在浏览某个网站时，从进入网站到浏览器关闭所经过的这段时间。

因此，利用 Session 可以管理用户的状态，并将一些数据与它进行关联。本节将详细介绍 PHP 中如何使用 Session 存储数据。

9.7.1 Session 概述

Session 解决了 HTTP 协议无法区分和保存客户端状态的难题，它可以为客户端用户分配一个编号。那么 Session 是如何分配的呢？它是通过 Session ID（简写 SID）来分配的。Session ID 是服务器端随机生成的 Session 文件的文件名，因此能够保证其唯一性进而确保 Session 的安全。

也就是说当一台 Web 服务器运行时，可能有若干个用户正在浏览这台服务器上的网站。当每个用户首次登录该网站时，服务器会自动的为这个用户分配一个 Session ID，来标识这个用户的身份。不同的用户会话信息会由不同的 Session 对象来保存，因此不必担心两个 Session 对象会发生冲突。图 9-19 所示的示意图说明了这个过程。

另外，在一般情况下，Session 对象是有生命周期的,即如果在规定的时间内没有对 Sesionn 对象进行刷新，系统将会终止这些对象。

9.7.2 向 Session 保存数据

在 PHP 中每一个 Session 都通过调用 session_start()函数开始，这个函数检查一个 Session 是否存在，如果不存在则创建一个新 Session。

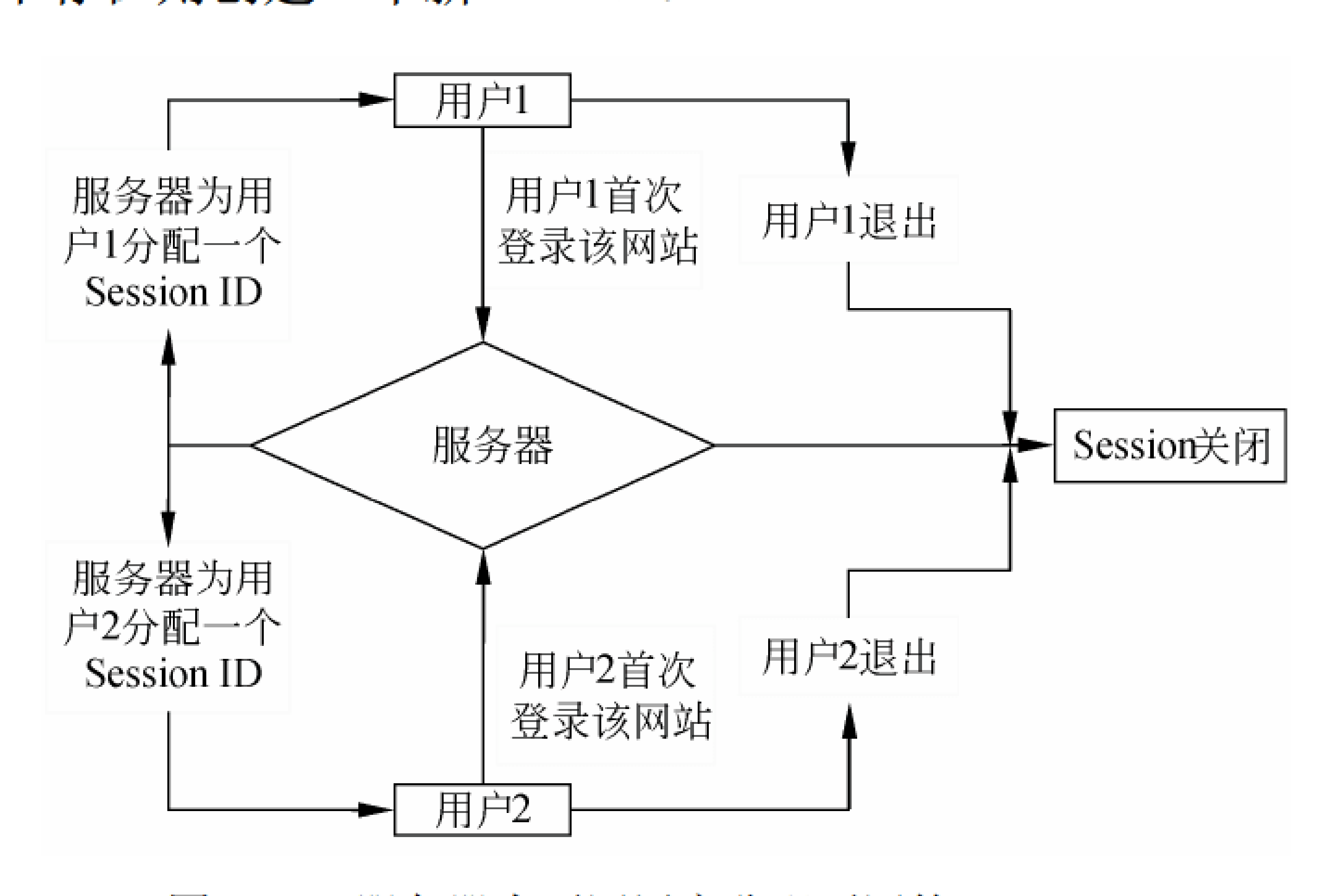


图 9-19 服务器为不同用户分配不同的 Session ID

第一次调用 session_start()时，将产生一个 Session ID，该 ID 将所有的会话数据绑定到某个特定用户。当然也可以通过调用 session_id()函数手动设置 Session ID，该函数的语法格式如下所示：

```
string session_id ( [string $id])
```

在调用时如果没有参数将返回当前 Session ID；如果提供了参数$id，则当前 Session ID 将被该值替换。

【实践案例 9-11】

创建一个案例，在页面中输出使用 session_id()函数前后 Session ID 的变化。创建一个

PHP 文件，然后添加如下的实现代码：

```
<h1>读取和设置 Session ID</h1>
<p>
    <?php  session_start(); ?>
    使用 session_id("myCustomSessionID")之前<br/>
    <?php echo "PHP 自动产生的 SID: <b>".session_id();?></b><br/><br/>
    使用 session_id("myCustomSessionID")之后<br/>
    <?php
    session_id("myCustomSessionID");
    echo "手动指定的 SID: <b>".session_id();
    ?></b>
</p>
```

在上述代码中第一次调用 session_id()函数时没有指定参数，此时将返回 PHP 自动产生的 Session ID。在第二次调用时指定当前 Session ID 为“myCustomSessionID”，最终运行效果如图 9-20 所示。

创建会话变量的形式如下：

```
$_SESSION[key] = value;
```



图 9-20　session_id()函数运行效果

其中，key 是字符串类型的变量，value 是需要保存到该 Session 变量中的值。例如，下面代码向 Session 中添加一个名为 pdname 的会话变量。

```
<?php
    $_SESSION['pdname'] = "新款 3G 智能机";        //创建名为 pdname 的会话变量
?>
```

要获得 Session 变量 pdname 中的值，也只需要使用$_SESSION['username']语句，如下所示：

```
<?php
    echo "产品名称: ".$_SESSION['pdname'];    //输出“产品名称：新款 3G 智能机”
```

```
?>
```

【实践案例 9-12】

使用 Session 创建一个简单的页面计数器。该计数器在用户第一次访问一个 Web 页面时初始化一个变量，以后每次用户重新访问这个页面时，该变量的值会自增 1。

创建一个 PHP 文件，然后添加如下的实现代码：

```
 <h2>
<?php
session_start();                        //创建 Session 变量开始一个会话
if(!isset($_SESSION['counter'])){
    $_SESSION['counter']=1;             //计数器设置为 1
} else{
    $_SESSION['counter']++;             //增加计数器
}
echo "您一共访问本页 <b>".$_SESSION['counter']."</b> 次。";
?></h2>
```

执行该段代码，首次访问该页面执行结果如下所示：

```
您一共访问本页 1 次。
```

如果再次访问该页面，页面计数器会自动增加访问次数，运行效果如图 9-21 所示。



图 9-21　计数器运行效果

如果用户在浏览器中新建一个窗口，重新访问这个页面，会发现计数器又从 1 开始。这就说明又与服务器建立了一个新的会话。

9.7.3　从 Session 读取数据

在 PHP 中可以使用 session_start()函数来开始 Session。session_start()函数的定义如下：

```
bool session_start ( void )
```

session_start()函数是创建一个新会话还是继续当前会话，取决于是否拥有 SID。开始会话时，只需使用如下形式调用 session_start()函数：

```
session_start();
```

在默认状态下，会话不会自动启动，但是可以修改 php.ini 文件中的 session.auto_start 参数的值，让会话自动启动。方法是在 PHP 安装目录中打开 php.ini 文件，在该文件中找到 session.auto_start 参数。

session.auto_start 参数的默认值为 0，这里将其值修改为 1，保存后重新启动 Apache 即可生效。

调用 phpinfo()函数可以查看 PHP 的配置效果。当 session.auto_start 参数的值为 0 时，PHP 配置信息如图 9-22 所示，当 session.auto_start 参数的值为 1 时，PHP 配置信息如图 9-23 所示。

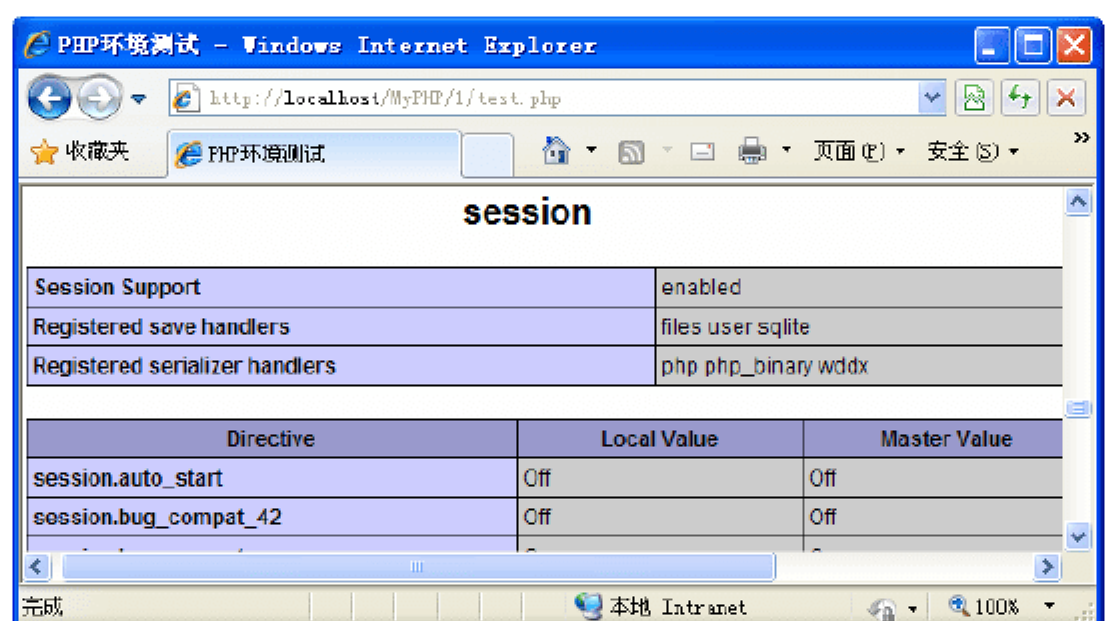


图 9-22　session.auto_start 参数值为 0 时

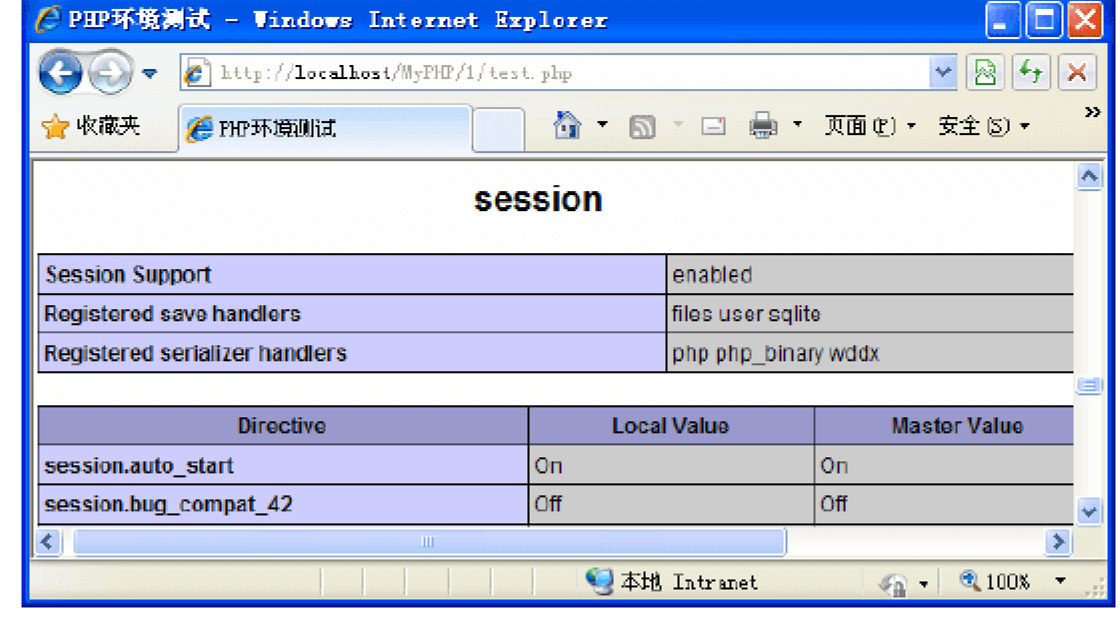


图 9-23　session.auto_start 参数值为 1 时

【实践案例 9-13】

使用 Session 功能实现一个简单的用户登录模块。要求：一个登录表单用于输入用户名和密码，提交之后由一个验证程序进行接收；在验证页面将用户名和密码保存到 Session 中，然后显示一个跳转链接；在链接的目标页面中获取保存到 Session 中的用户名和密码。

具体步骤如下所示。

（1）创建一个名为 login.php 的文件作为登录页面。

（2）在登录页面中使用 form 创建一个表单，再添加“用户名”和“密码”的文本框，以及“登录”按钮。这部分代码如下所示：

```
<form action="check.php" method="post">
    <table>
        <tr>
          <td><label for="login-username">账号：</label></td>
          <td><input type="text" name="username" vlaue="" autocomplete="off"
          id="login-username" class="input-box"></td>
        </tr>
        <tr>
          <td><label for="login-password">密码：</label></td>
          <td><input type="password" name="password" vlaue="" autocomplete=
```

```
          "off" id="login-password" class="input-box"></td>
        </tr>
           <tr>
             <td> </td>
             <td><input type="submit" value="立即登录" id="login-submit">
               <input type="button" value="注册" id="login-register"></td>
           </tr>
        </table>
      </form>
```

在上述代码中，要注意 form 的 action 属性和 method 属性，账号和密码文本框的 name 属性。

（3）创建一个名为 check.php 的文件，它接收 login.php 以 POST 方式提交的表单。

（4）在 check.php 中添加如下实现代码：

```
<?php
  if(!isset($_POST['username'])||!isset($_POST['password']))
  {
      echo "账号和密码都不能为空。<br/>";
      echo "<a href=\"login.php\" >单击这里重新登录。</a>";
  }else{
      $name = $_POST['username'];              //获取 username 参数的值
      $pass = $_POST['password'];              //获取 password 参数的值
      //session_start();
      $_SESSION['username'] = $name;           //创建 Session 变量保存账号
      $_SESSION['userpass'] = $pass;           //创建 Session 变量保存密码
  ?>
你输入的用户名是：<?php echo $_SESSION['username']; ?><br />
你输入的密码是：<?php echo $_SESSION['userpass']; ?><br />
<?php   echo "<a href=\"main.php\" >单击这里转向首页。</a>";
 }?>
```

如上述代码所示，在 check.php 文件中将接收的用户登录信息保存到 Session 变量中。然后在页面中输出 Session 变量中的信息，并提供一个超链接，链接到 main.php 页面。

（5）创建一个名为 main.php 的文件，它将作为登录之后的主页面显示保存在 Session 中的信息。

（6）在 main.php 的合适位置添加如下实现代码：

```
<?php
if(!isset($_SESSION['username'])){
    echo "<h3>没有权限访问此页面。</h3>";
    echo "<a href=\"login.php\" >单击这里重新登录。</a>";
    }
    else{
?>
```

```
    <h3> 欢迎会员: <?php echo $_SESSION['username']; ?> </h3>
    登录成功，您的密码是: <?php echo $_SESSION['userpass']; ?>。为了账号安全，请定
    期密码。
<?php }?>
```

在 main.php 页面中同样输出 Session 变量中的信息。如果能输出正确的信息，则可以证明 Session 会话变量具有存储值的效果。

（7）保存上面对文件的修改。确保 login.php、check.php 和 main.php 这 3 个文件在同一个目录下。在浏览器中通过 login.php 文件查看登录表单，如图 9-24 所示。

（8）填写用户登录信息，例如本示例中输入用户名为“ayhncn@gmail.com”，密码为 123456289。单击“登录”按钮，登录信息将提交到 check.php 页面中，该页面的运行效果如图 9-25 所示。



图 9-24 login.php

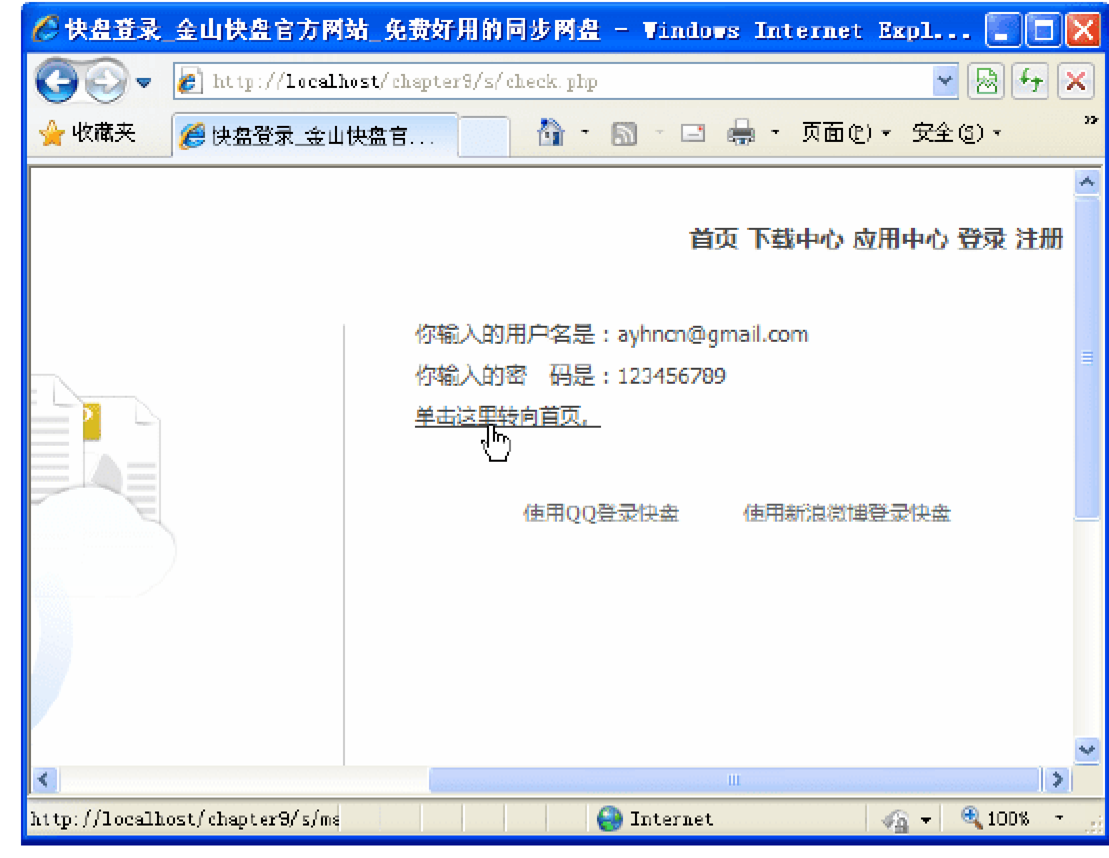


图 9-25 check.php 页面

（9）单击 check.php 页面中的超链接，访问 main.php 页面，效果如图 9-26 所示。可以看出，Session 会话变量确实起到了保存值的作用，在 main.php 页面中显示了在登录表单输入的用户名和密码。

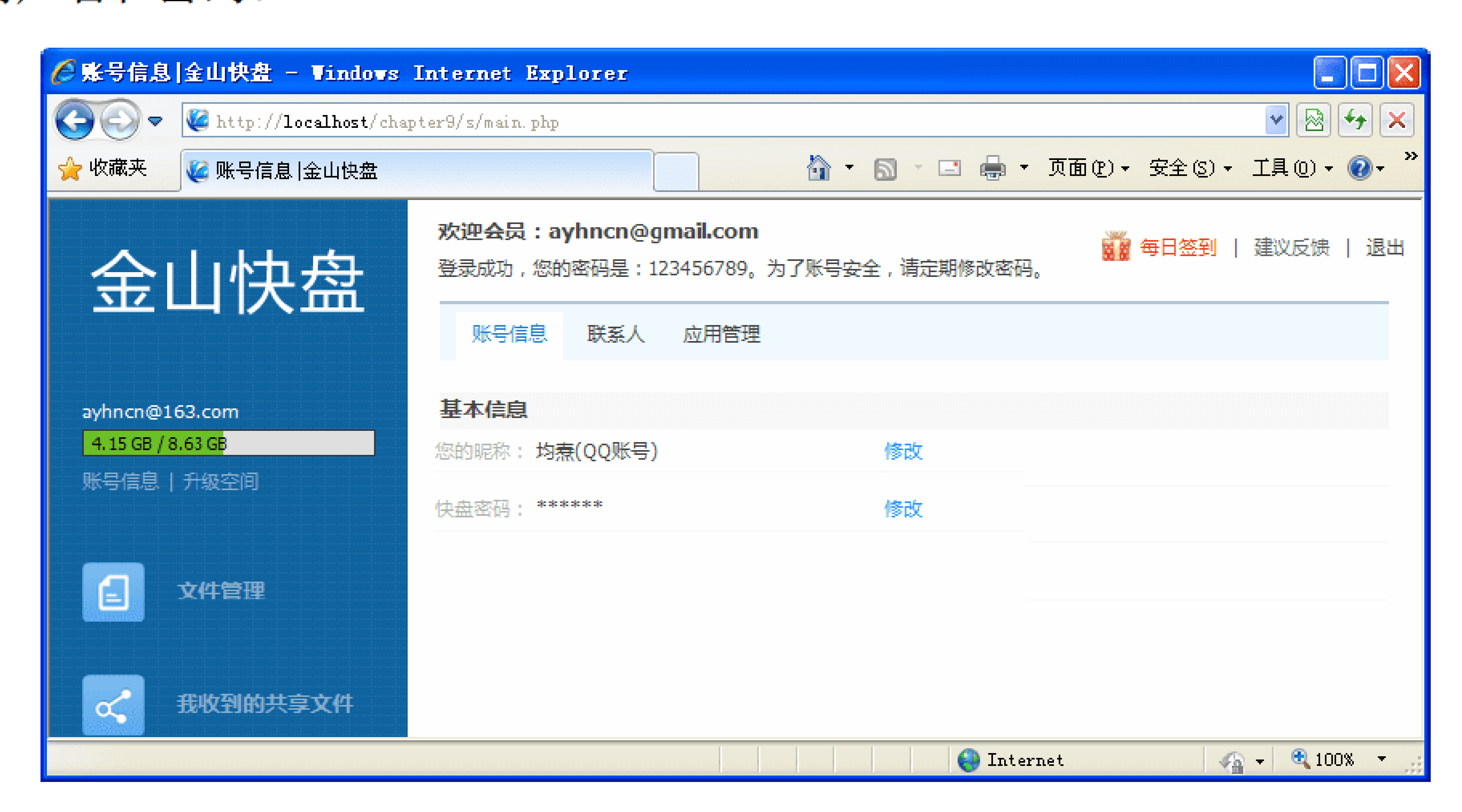


图 9-26 main.php 页面

9.7.4 删除 Session 数据

如果关闭浏览器，则当前会话自动中断，打开一个新的浏览器时，将打开一个新的会话。不过，在实际应用中，有时候也需要使用程序中断会话，或者删除会话变量，例如登录之后单击“安全退出”按钮，此时浏览器并没有关闭，所以应该手动将保存在 Session 变量中的用户登录信息删除。

删除会话变量需要使用 unset()函数，其使用形式如下：

```
unset($_SESSION[key]);
```

【实践案例 9-14】

对实践案例 9-13 登录后的首页进行修改，添加一个退出功能。实现方式其实很简单，就是将登录时创建的 Session 变量使用 unset()函数进行删除，具体步骤如下所示。

（1）打开 9.7.5 节的 main.php，在用户名后面添加一个退出链接。代码如下所示：

```
<h2> 欢迎会员：<?php echo $_SESSION['username']; ?> | <a href="exit.php">
退出</a></h2>
```

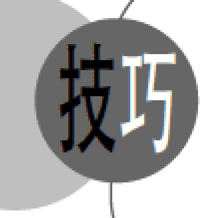
（2）如上述代码所示，退出功能由 exit.php 文件实现，这一步创建该文件。

（3）在 exit.php 文件中添加如下的主要代码：

```
当前会员信息：
<ul>
<li>会员名称：<?php echo $_SESSION['username']; ?></li>
<li>会员密码：<?php echo $_SESSION['userpass']; ?></li>
</ul>
正在退出。<br/>
<?php
unset($_SESSION['username']);                    //删除保存会员的变量
unset($_SESSION['userpass']);                    //删除保存密码的变量
?>
退出之后，会员信息如下：
<br />会员名称：<?php echo $_SESSION['username']; ?>
<br />会员密码：<?php echo $_SESSION['userpass']; ?>
```

在 exit.php 页面中首先输出 Session 变量的信息，然后使用 unset()函数删除 Session 变量，最后再次输出 Session 变量中的信息。

运行后的页面效果如图 9-27 所示。可以看到删除 Session 变量后，无法再获取保存在 Session 变量中的会员信息。



unset()函数可以删除一个 Session 中的数据。如果要一次删除当前 Session 中的所有数据可以调用 session_destroy()函数。如果要删除所有 Session 中的数据可以调用 session_unset()函数。



图 9-27　删除 session 会话变量后的效果

9.7.5　Session 数据的编码和解码

PHP 默认情况下是按顺序格式保存 Session 数据的，并且各个 Session 变量之间都会以分号隔开。每个数据都由 3 个部分组成：变量名称、值长度和值，存储语法格式如下所示：

```
变量名称|s:值长度:值;
```

例如，语句“$_SESSION["uid"] = "somboy"”执行后，如下所示为 Session 数据的内容：

```
uid|s:6:"somboy";
```

除了这种标准格式存储数据外，在 PHP 中还可以使用 session_encode()函数和 session_decode()函数手动完成会话数据的编码和解码，下面分别介绍这两个函数。

1. session_encode()函数

该函数可以使用手动方式将所有 Session 变量编码为一个字符串，语法格式如下所示：

```
string session_encode()
```

该函数返回值是编码后的字符串。

下面的示例代码首先向 Session 中存储数据，然后使用 session_encode()函数编码并输出编码后的字符串。

```
<?php
    session start();                         //开始 Session
    $ SESSION["system"] = "内部成绩管理系统 V1.0";
    $_SESSION["isauth"]=true;
    $_SESSION["client"]= "市中心一小实验学校";
    $ SESSION["tel"]="12345678";
    SESSION["admins"]=array(1=>"admin",2=>"netmini",3=>"system",4=>"zhht");
```

```
    $userString = session_encode();     //编码数据，将结果保存到$usetring 中
>
如下所示为使用 session_encode()函数后，Session 中的数据：<br/><br/>
<?php  echo $userString; ?>
```

执行上述代码，在页面会看到如下输出结果：


```
如下所示为使用 session_encode()函数后，Session 中的数据：

system|s:29:"内部成绩管理系统 V1.0";isauth|b:1;client|s:27:"市中心一小实验学校
";tel|s:8:"12345678";admins|a:4:{i:1;s:5:"admin";i:2;s:7:"netmini";i:3;
s:6:"system";i:4;s:4:"zhht";}
```

2. session_decode()解码函数

session_encode()返回的字符串可以通过 session_decode()函数解码，该函数的语法格式如下所示：

```
bool session_decode ( string $data )
```

该函数将$data 参数中的数据解码并返回最初的格式，如果执行成功则返回 true，否则返回 false。

下面的示例代码首先在 Session 保存一些数据，然后使用 session_encode()函数编码之后使用 Session_unset()函数删除 Session 中的数据。接下来，使用 session_decode()函数对编码的结果进行解码，再输出 Session 中的数据。

```
    <?php
      session_start();
       $_SESSION["sysname"] = "流量监控系统";   //保存数据到 Session 中
       $_SESSION["target"]="www.itzcn.com";   //保存数据到 Session 中
       $encode_str = session_encode();     //执行编码，结果保存到$encode_str
       echo"如下所示为使用 session_encode()函数后，Session 中的数据：<br/><br/>";
       echo $encode_str;                   //输出编码的结果
       session_unset();                    //删除 Session 数据
       echo "<hr/>删除之后，Session 中的数据丢失。<br/>";
       echo "系统名称：".$_SESSION['sysname'];//此时 Session 数据为空，无输出结果
       echo "<br/>";
       echo "网站地址：".$_SESSION['target'];//此时 Session 数据为空，无输出结果
       echo "<hr/>";
       session_decode($encode_str);        //解码$encode_str 的 Session 数据
       echo "解码后 Session 中的数据恢复。<br/>";
       echo "系统名称：".$_SESSION['sysname'];      //输出结果
       echo "<br/>";
       echo "网站地址：".$_SESSION['target'];       //输出结果
  ?>
```

执行上述代码，在页面会看到如下输出结果：

```
sysname|s:18:"流量监控系统";target|s:13:"www.itzcn.com";
--------------------------------------------------------------------
删除之后，Session 中的数据丢失。
系统名称：
网站地址：
-----------------------------------------------------------------------
解码后 Session 中的数据恢复。
系统名称：流量监控系统
网站地址：www.itzcn.com
```

9.8 文件上传

文件上传是 Web 中最常见的应用之一。在 PHP 中可以接受任何来自标准浏览器的上传文件，使用这种特性可以上传文本文件、图片或者二进制文件。本节将详细介绍上传文件时表单的准备工作以及在文件上传后进行的处理。

9.8.1 准备文件上传表单

与普通的表单提交数据不同，在实现文件上传时，表单的 method 方式必须为 POST，否则无法实现文件上传功能。另外，还需要添加上传的属性 enctype="multipart/form-data"，该属性指示浏览器可以提供文件上传功能，服务器端提交的数据中包含文件的数据。

如下代码即为一个支持文件上传的表单代码：

```
<form name="form1" method="post" action=" " encte="multipart/form-data">
    <input type="hidden" name="max file size" value="264104"/>
    您要上传的文件：<input type="file" name="file" id="file"/>
    <input type="submit" name="upload" id="button" value="提交"/>
</form>
```

它与普通表单的不同之处主要有如下几点。

- 表单使用 POST 方式进行提交，并且有一个 enctype 属性提示表单中有二进制文件数据。
- type 属性为 file 将显示一个文件输入框，并提供“浏览”按钮允许用户选择文件。
- type 属性为 hidden 表示隐藏域，通过其 value 值指定允许上传文件的最大尺寸（以字节为单位）。



这里的隐藏域并不能真正地限制文件上传的大小，而是将它作为一个变量的值随表单一起提交，然后在 PHP 端进行比较和验证。

9.8.2 处理上传文件

用户通过客户端浏览器的上传表单提交之后，PHP 将会自动生成一个$_FILES 数组，其中保存了上传文件的信息。

假设选择文件的代码如下：

```
<input type="file" name="file" id="file"/>
```

那么，关于该文件的所有信息都包含在$_FILES["file"]数组中，并且在该数组中包含了如下的键。

- $_FILES["file"]["name"] **被上传文件的名称，例如** b.jpg、a.png。
- $_FILES["file"]["type"] **被上传文件的类型，例如** image/png。
- $_FILES["file"]["size"] **被上传文件的大小，以字节为单位。**
- $_FILES["file"]["tmp_name"] **存储在服务器的文件的临时副本的名称。**
- $_FILES["file"]["error"] **由文件上传导致的错误代码。**

在文件上传时，$_FILES["file"]["error"]会返回不同的常量值表示不同的错误，如表 9-4 所示。

表 9-4 文件上传错误

返回值	说明
UPLOAD_ERR_OK	没有错误发生，文件上传成功
UPLOAD_ERR_INI_SIZE	上传文件大小超过 php.ini 中 upload_max_filesize 选项限制的值
UPLOAD_ERR_FORM_SIZE	上传文件大小超过表单中 max_file_size 选项指定的值
UPLOAD_ERR_PARTIAL	文件只有部分被上传
UPLOAD_ERR_NO_FILE	用户没有提供任何文件上传（没有选择文件）

如下是一个完整的 PHP 端接收文件、文件检查以及上传的示例代码：

```
<?php
if(isset($_POST["upload"]))                              //单击“提交”按钮
{
    //定义允许上传文件类型数组
    $allowtype=array("image/gif","image/jpeg","image/png","image/jpg","i
    mage/pjpeg");
        if($_FILES["file"]["size"] == 0)                  //发生错误
            echo("错误，不能读取文件。<hr/>");
        if ( $_FILES["file"]["error"] == 2)               //检测文件大小
            echo "错误，文件不能超过".$_POST["max_file_size"]." 字节。<hr/>";
        if(!in array($ FILES["file"]["type"], $allowtype)) {
                                                          //检测文件类型
            echo("错误，不支持当前的文件类型，请重新选择。<hr/>");
        }
        else{
            echo "<table width=100%><tr><td><h2>文件上传成功,信息如下</h2>";
```

```
            echo "文件名称: " . $_FILES["file"]["name"] . "<br />";
                                                          //上传的文件名称
            echo "文件类型: " . $_FILES["file"]["type"] . "<br />";
                                                      //上传的文件类型
            echo "文件大小: " . ($_FILES["file"]["size"] / 1024) . " Kb<br
/>";                                                  //文件大小
            echo"文件临时副本名称: ".$_FILES["file"]["tmp_name"]."<br/>";
                                                      //文件别名
            if (file_exists("uploads/" . $_FILES["file"]["name"])){
                                                      //判断文件是否存在
                echo $_FILES["file"]["name"] . " 文件已经存在. ";
                                                      //输出存在相同文件提示
            }
            else{
                move_uploaded_file($_FILES["file"]["tmp_name"],//开始上传
                "uploads/" . $_FILES["file"]["name"]);
                echo "文件存储在: " . "uploads/" . $_FILES["file"]["name"];
                                                      //输出上传后的路径
            }
            echo "</td><td>";
            echo "<img src="."uploads/" . $_FILES["file"]["name"]."
width=100>";                                          //显示图书
            echo "</td></tr></table>";
        }
    }
    ?>
```


将上述代码与 9.8.1 节的表单保存到一个 PHP 文件中，然后运行。首先选择一个图片类型文件单击“提交”按钮上传成功将看到如图 9-28 所示的效果。如果上传失败将看到提示信息，如图 9-29 所示。



图 9-28 文件上传成功



图 9-29 文件上传失败

9.9 文件下载

与文件的上传相比，下载文件要简单得多。首先需要用户单击一个链接触发下载动作，示例代码如下：

```
<a href="?action=download">下载文件到本地</a>
```

然后还需要指定要下载文件的名称和路径，并打开该文件输出文件类型、大小和内容，代码如下：

```
header("Content-type:application/octet-stream");        //文件类型
header("Accept-Ranges:bytes");                           //文件大小单位为字节
header("Accept-Length:".filesize('1.jpg'));              //文件大小
header("Content-Disposition:attachment;filename=1.jpg");
                                                  //以附件形式指定下载文件的名称
```

最后使用 PHP 中的 fread()函数将文件内容直接在页面中输出，让浏览器提示用户下载。所有的这些处理都是在服务器端完成的，因此用户是不会知道文件具体位置信息的，是非常安全的一种下载方法。

【实践案例 9-15】

编写一个案例，实现在页面上单击一个链接可下载文件的功能，具体步骤如下所示。

（1）首先创建一个 FileDownload.php 文件作为实例文件。

（2）在文件中编写一个文件下载的列表，其中包含下面的代码：

```
<table width="400" align="center">
 <tr>  <th bgcolor="#CCCCCC">文件名称</th>  <th bgcolor="#CCCCCC">操作</th>
 </tr>
<tr>
  <td>网站 Logo 图片</td>
  <td><a href="download.php?action=download&fname=logo.jpg">下载</a>
  </td>
</tr>
<tr>
  <td>第 2 批代理商审批结果.xls</td>
  <td ><a href="download.php?action=download&fname=agent_2.xls">下载
  </a></td>
</tr>
<tr>
  <td>订购单.doc</td>
  <td><a href="download.php?action=download&fname=<?php echo urlencode
  ("订购单.doc");?>">下载</a></td>
</tr>
</table>
```

可以看到当单击“下载”链接将向 Download.php 文件传递两个参数，action=download 表示要执行下载动作，fname 表示要下载的文件名称。

（3）在文件 FileDownload.php 所在的目录下新建 Download.php 文件。

（4）在 Download.php 文件中编写真正实现下载的代码，这部分代码如下所示：

```
<?php
if(isset($_GET["action"]))                    //是否单击“下载”链接
{
    $file_name = urldecode( trim($_GET["fname"]));      //获取要下载的文件名
    $file_dir = "uploads/";
    $fileurl=$file_dir . $file_name;          //指定文件路径
    if (!file_exists($fileurl)) {             //检查文件是否存在
        echo "文件找不到";                      //输出错误提示
        exit;                                 //退出
    } else
    {
        $file = fopen($fileurl,"r");          //打开文件
        Header("Content-type: application/octet-stream");
                                              //输入文件类型
        Header("Accept-Ranges: bytes");
        Header("Accept-Length: ".filesize($fileurl));  //输入文件大小
        Header("Content-Disposition: attachment;filename=" . $file_name);
                                              //输入文件名称
        echo fread($file,filesize($file_dir . $file_name));//开始下载
        fclose($file);                                 //关闭文件
        exit;                                          //退出
    }
}
?>
```

（5）浏览 FileDownload.php 文件，从文件列表中单击“下载”链接之后则弹出一个文件下载对话框，效果如图 9-30 所示。

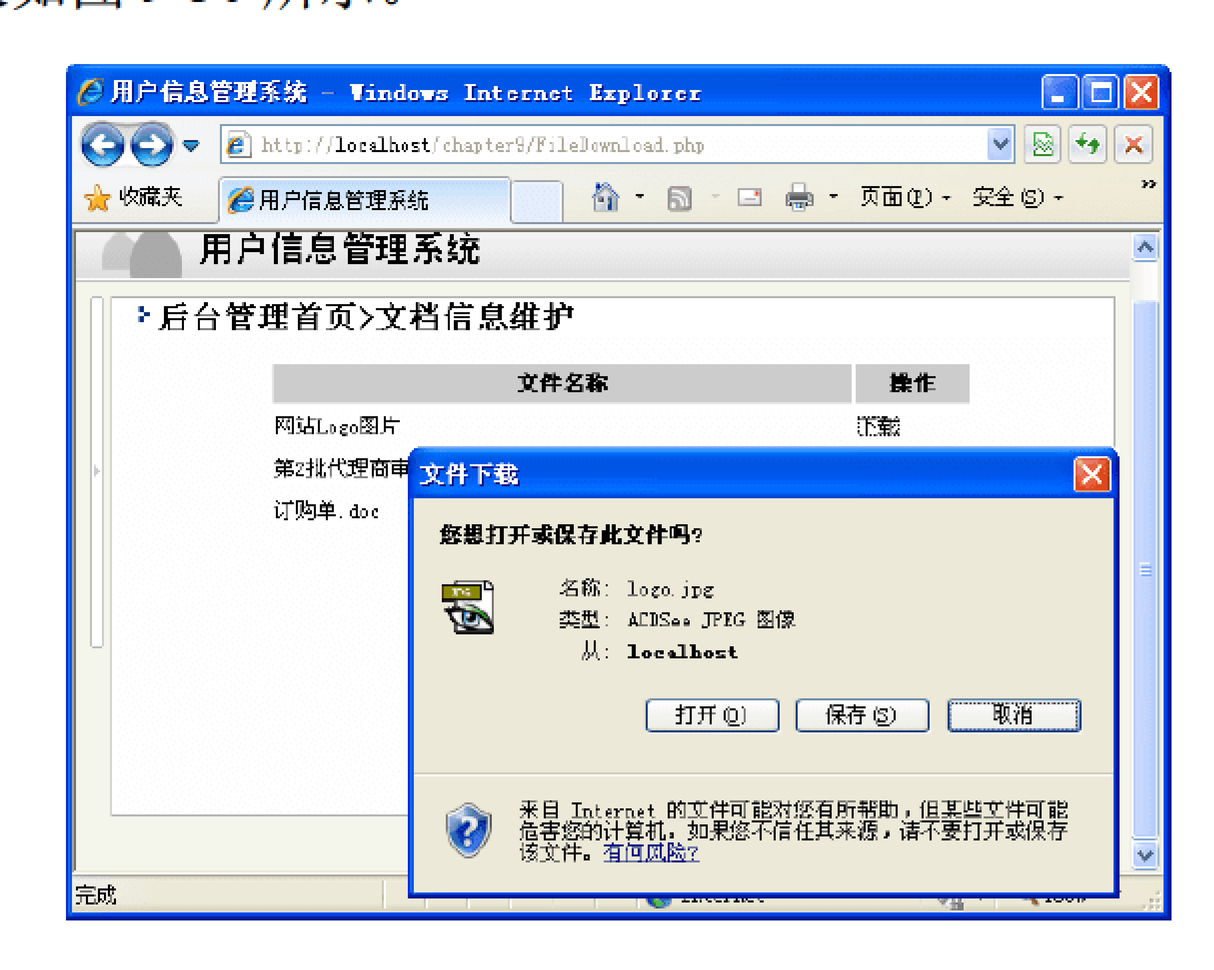


图 9-30　文件下载

9.10 项目案例：制作简单留言本

对于经常上网的读者来说，对留言本一定不陌生。留言本是一个和浏览者交流的平台，当浏览者留下意见之后，这些内容除被保存到服务器中外，还能够以 Web 形式显示到客户端浏览器上。

留言本可以设计得很简单，仅记录浏览者的意见，也可以很复杂，简单与否完全由开发者的能力及具体情况而定。本案例是一个简单型的留言本，具有如下功能。

- **留言模块　普通用户（游客）在浏览网站时可以查看所有留言，并发表自己的留言。**
- **管理员模块　管理员需要登录才能进行管理操作，完成后可以退出。**
- **管理留言　对用户发表的留言进行删除操作，此操作只有在管理员登录后才可以进行。**

【实例分析】

根据上面的描述，保存留言信息是实现的难点，因为它不仅需要保存到服务器端，而且需要能长期保存，且针对所有用户都有效。同时，如果管理员登录之后还必须可以能够它进行删除操作。实现这一点最佳的解决方案是使用数据库，但是由于还没有学习，在这里使用 Cookie 来代替。在下一章学习数据库之后将会对此案例进行重写。

要判断管理员是否已经登录可以通过检测 Session 中是否有值来实现，具体的实现步骤如下。

（1）新建一个 new.html 作为发表留言页面。然后添加一个表单允许用户填写自己的个人信息、留言内容和标题等。

```
<h1><img src="images/edit-icon.gif"/>发表新留言</h1>
<form action="dao.php?do=new" method="post" id="form1" name="form1"
onsu"return check();">
  <dl>
    <dt>
      <label for="title">标题：</label>
      <input type="text" width="260px" name="title"  id="title"/>  (×必填>
    <dt>
      <label for="vname">姓名：</label>
      <input type="text" width="260px" name="vname"  id="vname"/> (×必填>
    <dt>
      <label for="url">邮箱：</label>
      <input type="text" width="260px" name="url"  id="url"/>
    </dt>
    <dt>
      <label for="body">内容：</label>
      <textarea cols="40" rows="5" name="body"  id="body"></textarea>
    </dt>
```

```
    <dd> (×必填) </dd>
  </dl>
  <input type="submit" id="btnsave" name="btnsave" value="保存"
  styldth:60px;" />
  <input type="reset"  id="btn" name="btn" value="重置"  styldth:60px;"/>
</form>
```

创建好表单后，在页面中浏览，会看到标题、姓名、网址和内容 4 个文本框及“保存”和“重置”按钮。要实现发表效果，需要在单击“保存”按钮后获取用户在表单输入的信息，再将数据保存到 Cookie 中。注意上面表单虽然使用的是 POST 提交方式，但是仍然可以在 action 指定的 URL 中传递 GET 参数，这里的 do=new 表示从发表留言页面提交。

（2）根据上面的代码，在 new.html 文件所在目录创建一个名为 dao.php 的文件。使用 dao.php 文件实现向 Cookie 中保存留言信息，这部分实现代码如下所示：

```
<?php
$dos = array('new', 'delete');          //允许动作
$do = (!empty($_GET['do']) && in_array($_GET['do'], $dos))?$_GET['do']:
'index';
//保存留言操作
if($do=='new') {
    $title=$_POST["title"];                  //标题
    $visitor=$_POST["vname"];                //姓名
    $url=$_POST["url"];                      //网址
    $body=$_POST["body"];                    //内容
    $create_at=date("Y-m-d H:i:s");          //时间
    if($title=="" or $visitor=="" or $body==""){        //验证是否为空
        showmsg("new.html","标题、姓名和内容都不能为空，请检查。");
    }else{
        //创建一个表示留言的数组
        $m=array("t"=>$title,"v"=>$visitor,"b"=>$body,"d"=>$create_at,
        "u"=>$url);
        if(!empty($_COOKIE['messages']))    //判断当前 Cookie 中是否有留言信息
        {
            $messages=array();               //创建一个临时数组
            $messages=unserialize($_COOKIE['messages']);    //获取所有留言
            array_push($messages, $m);       //向数组中添加一个新元素
            setcookie("messages",serialize($messages));     //保存数组
        }else{                      //第一次运行时没有留言，此时创建一个空数组
            setcookie("messages",serialize(array($m)),time()+864000);
        }
        showmsg("index.php","添加成功，转到首页。");
    }
}
?>
```

上述代码的执行流程是，首先判断是否要执行保存留言操作。如果满足$do=='new'条件，则再获取输入的每个信息，接下来判断所需的必填项是否都满足，否则给出提示。输入完整后在当前的 Cookie 中进行查找，判断是否有键为 messages 的值，如果没有说明是第一次运行，此时创建一个数组来保存留言；否则则需要向当前的数组中插入新元素。最后给出成功提示，如图 9-31 所示。



图 9-31　发表留言



showmsg()是一个用户自定义的辅助函数，保存在 include\functions.php 文件中。限于篇幅关系，这里不再给出函数具体实现。

（3）创建 index.php 页面作为留言的首页，在这里会显示所有的留言信息。具体代码如下所示：

```
<h1><img src="images/edit-icon.gif"/>查看所有留言</h1>
<?php
if(!empty($_COOKIE['messages']))        //如果 Cookie 中有留言信息
{
    $messages=unserialize($_COOKIE['messages']);        //获取留言数组
    foreach ($messages as $m) {         //遍历数组
?>
<h2>标题: <?php echo $m["t"];?> </h2>
   <div id="cls" class="cls">  <?php echo $m["b"];?></div>
      <p class="date">Posted by <?php echo $m["v"];?> <img src="images/
      more.gif" alt="" /> <?php echo $m["u"];?> <img src="images/
      comment.gif" alt="" /> 留言于<?php echo $m["d"];?> <img src=
      "images/timeicon.gif" alt="" /> </p>
   <br/>
<?php
    }
```

```
}else{                                          //输出内容为空
    echo " <h2>当前没有留言 </h2>";
}
?>
```


上述代码比较简单，主要是通过遍历 Cookie 中的留言数组来输出留言信息。运行效果如图 9-32 所示。



图 9-32　查看留言

（4）经过前面 3 步留言的发表、保存和显示就实现了。下面创建一个 login.html 文件作为管理员登录页面，包含的代码如下所示：

```
<h1><img src="images/edit-icon.gif"/>管理员登录</h1>
<table width="392" height="96" border="0" cellpadding="0" cellng="0">
  <form name="form1" method="post" action="loginCheck.php">
    <tr>
      <td width="138" height="35">用户名:
        <input name="user" type="text" id="user3" size="15" valumin"></td>
    </tr>
    <tr>
      <td height="29">密    码:
        <input name="pass" type="password" id="pass2" size="15" valumin">
        </td>
    </tr>
    <tr>
      <td height="32"><input type="hidden" name="do" id="do" value="login">
        <input type="submit" value="登录 " style="width:60px;"/>
        <input name="重置" type="reset" value="重置 " styldth:60px;"/></td>
    </tr>
  </form>
</table>
```

（5）login.html 的信息会提交到 loginCheck.php 页面，创建该文件。然后编写代码，根据输入的用户名和密码与系统内置用户名和密码进行比较，如果相同说明登录成功则保存到 Session 中，否则给出错误提示。具体实现代码如下所示：

```
<?php
if($_POST["do"]=="login")                          //单击了“登录”按钮
{
    $user=trim($_POST['user']);
    $pass=trim($_POST['pass']);
    if($user==ADMIN_USER && $pass==ADMIN_PASS){    //判断该用户和密码是否正确
        echo "登录成功!";
        $_SESSION['adminuser']=$user;
        $_SESSION['adminpass']=$pass;
        echo "<meta http-equiv=\"refresh\" content=\"3;url=admin.php\">3
        秒钟转入主页,请稍等......";
    }else{
        echo "<script>alert('登录失败!');history.back();</script>"; }
}
?>
```

上述代码中的 ADMIN_USER 和 ADMIN_PASS 是两个常量，它们定义在 functions.php 文件中。管理员登录表单的运行效果如图 9-33 所示。



图 9-33 管理员登录

（6）如果登录成功将会转到管理留言页面 admin.php。创建该文件并在顶部添加如下的代码进行确认已经登录：

```
<?php
include "include/functions.php";
if(!isset($_SESSION['adminuser'])||empty($_SESSION['adminuser'])){
                                            //如果没有登录
```

```
    showmsg("login.html","请先登录再操作。");
}
?>
```

这是因为 admin.php 页面可以对留言进行删除操作。为了保护留言信息的安全，需要再次判断，从而防止了用户直接访问 admin.php 页面删除留言的问题。

（7）在如图 9-34 所示的管理留言页面有一个“退出”链接，它的地址是 loginCheck.php?do=exit。对应的实现代码如下所示，需要添加到 loginCheck.php 文件中。

```
if(isset($_GET["do"])&&($_GET["do"]=="exit")){
    $_SESSION["adminuser"]=null;
    $_SESSION["adminpass"]=null;
    echo "<script>alert('退出成功!');history.back();</script>";
}
```



图 9-34　管理留言

（8）最后来看一下管理留言页面中删除留言的链接，代码如下：

```
<a href="dao.php?do=delete&id=<?php echo $index;?>">删除</a>
```

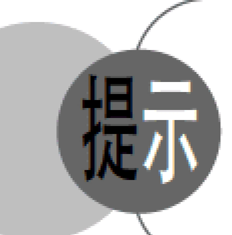
可以看到单击之后将转到 dao.php 文件，并传递两个参数 do=delete 表示执行删除操作；id 参数表示要删除留言的索引。

（9）在 dao.php 中编写从 Cookie 中删除指定索引的留言信息，实现代码如下所示：

```
//删除操作
if($do=='delete') {
    $id=$_GET["id"];                              //获取索引
    if(!empty($id)){
        if(!empty($_COOKIE['messages'])){         //如果留言不为空
            $messages=array();
            $messages=unserialize($_COOKIE['messages']);
```

```
                                                      //获取留言信息数组
                array_remove($messages,$id);          //从中删除元素
                setcookie("messages",serialize($messages));//再次保存
                showmsg("admin.php","删除成功，转到首页。");
            }else{
                showmsg("admin.php","删除成功，转到首页。");
            }
        }
    }
```

（10）再次运行 admin.php 页面，当单击“删除”链接之后会提示删除成功并刷新页面。此时会发现留言列表发生了变化。



出于篇幅关系，这里没有给出本案例的所有实现代码。读者可从本书随书光盘获取完整源代码。

9.11 习题

一、填空题

（1）HTML 表单的__________属性用于指定提交方式，默认值为 get。

（2）在下面代码的空白处填写代码，使程序可以显示“春天里”。

```
<?php   $song="春天里";    ?>
正在播放《<input name="name" type="text" id="name" value="__________" />》
```

（3）假设在页面的表单中包含如下代码，现在要获取用户的选择结果应该使用代码__________。

```
<input type="radio" name="season" id="spring" value="春" />春
<input type="radio" name="season" id="summer" value="夏" />夏
<input type="radio" name="season" id="autumn" value="秋" />秋
<input type="radio" name="season" id="winter" value="冬" />冬
```

（4）假设有下面一段 PHP 代码，在空白处填写合适的代码，使程序可以实现 Cookie 的删除。

```
<?php
setcookie("web"," www.baidu.com");
echo("当前 web 的值 ".$_COOKIE['web']."<br>");          //输出当前值
__________;                                             //删除 Cookie
?>
```

二、选择题

（1）下面选项中不属于表单元素的是____________。

A．fieldset

B．checkbox

C．reset

D．textarea

（2）假设有个表单提交后的 URL 地址如下，那么使用代码__________可以获取到值 bbs。

```
http://localhost/chapter9/search.php?tid=356&mod=bbs&goto=index
```

A．$_GET["mod"]

B．$_POST["mod"]

C．$_COOKIE["mod"]

D．$_SESSION["mod"]

（3）使用下面的____________选项可以获取表单提交前的页面 URL 地址。

A．$_SERVER['REQUEST_METHOD']

B．$_SERVER['SERVER_NAME']

C．$_SERVER['PHP_SELF']

D．$_SERVER['HTTP_REFERER']

（4）当 Cache-Control 的值为____________时表示请求和响应不缓存。

A．public

B．private

C．no-store

D．no-cache

（5）使用文件上传时必须将____________属性设置为 multipart/form-data。

A．method

B．action

C．enctype

D．target

（6）下面关于 Session 使用的代码，不正确的是__________。

A．$_SESSION['web'] = "baidu.com";

B．unset($_SESSION["web"]);

C．session_unset("web");

D．echo $_SESSION['web'];

（7）在$_FILES 数组中使用__________键可以获取被上传文件的类型。

A．name

B．type

C．size

D．tmp_name

三、上机练习

1．制作评论发表和显示

本次上机实践要求读者制作一个评论发表表单，然后使用 GET 方式进行提交并输出获取的结果。评论表单包含的内容及运行效果如图 9-35 所示。

2．实现网站后台登录和首页

本次上机实践要求读者首先制作一个登录页面，其中包含登录名、密码和验证码，如图 9-36 所示。验证码可以使用 PHP 的随机函数生成，然后将数字保存到 Cookie 中。在登录表单提交时对登录名、密码和验证码都进行检测，参考效果如图 9-36 所示。

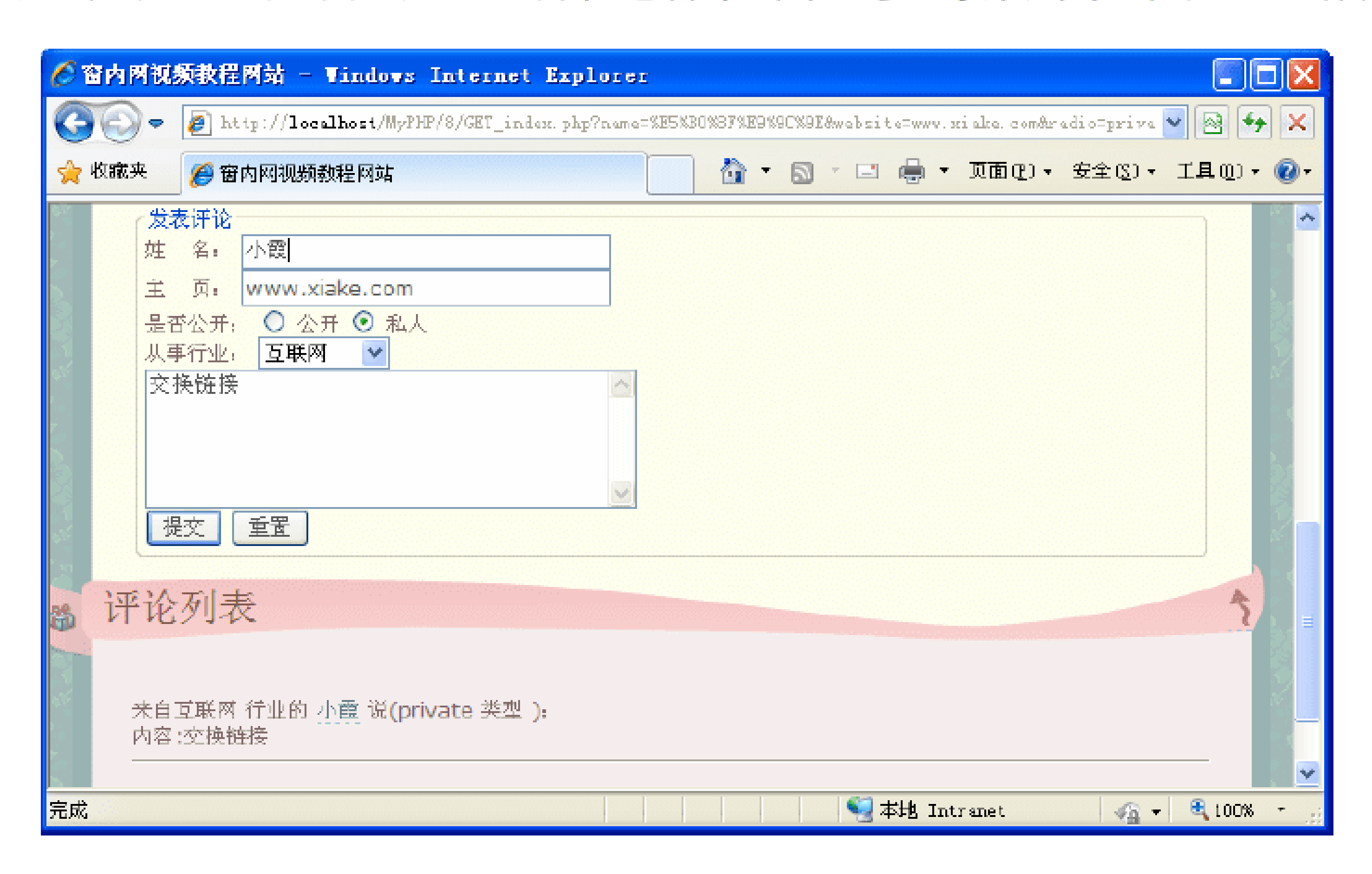


图 9-35　评论功能



图 9-36　登录页面



图 9-37　管理首页

登录成功之后转到管理首页，在这里显示了登录的用户名以及退出功能，如图 9-37 所示。

9.12 实践疑难解答

9.12.1 关于表单提交的问题



关于表单提交的问题

网络课堂：http://bbs.itzcn.com/thread-19694-1-1.html

【问题描述】：表单非常简单，只有两个文本框和一个按钮。想实现提交表单时，如果两个文本框是空则不提交表单。我用的是 return，但是表单还是提交，这是为什么？请教各位，谢谢了。

【解决办法】：这是网站开发时经常用的功能，对表单数据进行有效性检查。如果不符合条件就不用提交。这需要用到 form 的 onsubmit 事件，如果它的值为 false 就不用提交。因此，可以编写一个函数进行判断，把不符合条件的代码写在函数里并 return false。

示例代码如下：

```
<form name="formsearch" id="formsearch" action='/search.php' method="post"
onsubmit="return Check()">
<input name="searchword" type="text" value="输入影片名或演员名" onClick=
"if(this.value=='输入影片名或演员名')this.value=''" id="searchword"/>
<input type="submit" value="搜索"/>
</form>
<script type="text/javascript">
function Check(){
    var sw=document.getElementById("searchword").value;
    if(sw=="") return false;
    if(sw=="输入影片名或演员名")  {
        return false;
    }
    if(sw.length>20){
        alert("输入更少内容进行搜索");
        return false;
    }
}
</script>
```

9.12.2 表单验证 JavaScript 和 PHP 哪个消耗的数据流量更大



表单验证 JavaScript 和 PHP 哪个消耗的数据流量更大

网络课堂：http://bbs.itzcn.com/thread-19695-1-1.html

【问题描述】：假设，在我的程序中有一个 form 表单，里面包含了很多输入项。现在问

题是要对它们进行验证，可行方案有如下两种。

- **通过 JavaScript 对表单里面的 input 进行验证。**
- **通过 PHP 接收数据并验证。**

那请问这两种验证方法哪个消耗的数据流量更大呢?

【解决办法】：严格来讲肯定是 PHP 消耗的数据量大，不过这个数据是可以忽略的。这个问题的重点是何时使用 JavaScript 或者 PHP 进行表单的验证。

从执行方式来讲，JavaScript 在客户端执行适合对用户体验有要求的情况下使用；而 PHP 运行在服务器端需要每次提交到后台，这也可以做更多验证，像比较数据库的情况。

根据本人的经验，在 PHP 项目中使用表单验证的顺序如下。

（1）用 JavaScript 验证输入是否合法。例如：不能为空、必须为 5~12 个字符、是否是有效邮箱格式之类的。

（2）用 ajax 请求判断数据是否已存在。例如：用户名、邮箱已存在等。

（3）最后提交的时候用 php 验证。例如：数据是否合法，这是最后一道验证。

9.12.3 session_destroy()的问题



session_destroy()的问题

网络课堂：http://bbs.itzcn.com/thread-19696-1-1.html

【问题描述】：代码如下：

```
<?php
  session_start();
  $_SESSION['name']='liu';
  session_destroy();
  echo $_SESSION['name'];
?>
```

我的 session_destroy()怎么删除不了 Session 呢，是不是要修改配置文件。

【解决办法】：

session 和$_SESSION 是两回事。你应该使用 unset($_SESSION['name'])函数来注销$_SESSION['name']。

session_unset()用于释放当前在内存中已经创建的所有$_SESSION 变量，但不删除 session 文件以及不释放对应的 sessionid。

session_destroy()用于删除当前用户对应的 session 文件以及释放 sessionid，内存中的$_SESSION 变量内容依然保留。

所以，一般要两个一起配合起来用。

9.12.4 文件下载的实现



文件下载的实现

网络课堂：http://bbs.itzcn.com/thread-19697-1-1.html

【问题描述】：我是 PHP 菜鸟哈，刚学 PHP 没多久，现在在做一个能够上传下载文件的文件管理系统，要求就是用户能够上传文件，并能够下载上传的文件。

现在文件上传已经实现了，并且上传文件也能够在列表中显示出来，就是不知道下载到底怎么实现。

【解决办法】：这里假设楼主上传文件后将文件路径保存在数据库，表的结构如下：

```
file_id(自增 ID)        path(文件上传后的路径)
  1                  upload/temp.rar
```

那么下载有如下两种方式可以实现。

（1）将 A 标签的 href 属性直接指向文件，即用如下形式：

```
<a href="http://www.domain.com/upload/temp.rar">下载标题</a>。
```

这种形式用户单击之后服务器会自动将该文件发送到浏览器给用户提供下载，但是可执行脚本是不能够下载的，比如上传文件是 temp.php，那么你访问的时候如果 upload 文件夹有执行权限则 temp.php 会作为服务器端的脚本来执行。当然，一般情况下，为了服务器的安全上传文件夹是不具备执行权限的，而且 php 文件一般也是不接受的。

缺点就是，用户直接下载文件，服务器端无法再做更多的处理，比如记录下载次数等。

（2）将 A 标签的 href 属性指向一个 PHP 程序，由该程序接受请求后设置 http 标头信息然后读取文件并输出提供给用户下载。例如：

```
<a href="http://www.domain.com/upload/download.php?file_id=1">下载标题</a>
```

这时 download.php 程序接收到一个参数 file_id=1，于是就可以根据这个参数来查询数据库获得目标文件的路径，获得到路径后设置 http 标头信息并且读取文件输出，下载开始。

这种方式安全性要远好于上一种，不仅隐藏了服务器的路径信息，灵活性也比较高，更重要的是它可以让我们进行更多的处理，比如记录文件下载次数、记录当前用户已经下载过该文件之类的。

示例程序：

```
//文件下载函数，接受参数为文件的路径
function download($file)
{
    $fp= fopen($file, "r"); //打开文件
    $fileSize=filesize($file);
    //输入文件标签
    $fileInfo=pathinfo($file);
    Header("Content-type: application/octet-stream");
    Header("Accept-Ranges: bytes");
    Header("Accept-Length: $fileSize");
    Header("Content-Disposition: attachment; filename=" . $fileInfo
```

第 10 章 MySQL 数据库与 PHP 处理

要开发一个动态网站，数据库的使用是必不可少的。PHP 几乎支持所有的数据库，与其中的 MySQL 数据库更是最佳组合，可以充分发挥两者的优势。

本章以 MySQL 数据库为例，首先介绍有关它的简单操作。然后向读者介绍 PHP 连接 MySQL 的两种方式 mysql 和 mysqli，接下来则对每种方式的具体操作进行详细讲解。像如何连接数据库、关闭连接、执行查询、获取数据、显示错误以及 MySQL 的信息等。

本章学习要点：

- 掌握 MySQL 数据库的安装和配置
- 掌握 mysql MySQL 库连接和关闭的方法
- 掌握 MySQL 执行查询语句和获取结果显示的方法
- 熟悉获取 MySQL 数据库信息的方法
- 掌握 mysqli 库执行查询的方法
- 掌握 mysqli 库获取结果集的过程
- 熟悉 mysqli 库中预处理语句的使用

10.1 MySQL 数据库

MySQL 是一个开放源码的小型关系型数据库管理系统，由 MySQL AB 公司（被 Oracle 收购）开发。目前 MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，很多型网站选择 MySQL 作为数据库，像 Yahoo 和 Google 等。

10.1.1 安装 MySQL 数据库

在使用 MySQL 之前必须先进行安装，可以从 MySQL 的官方网站 http://www.mysql.com 找到关于 MySQL 的最新版本信息。下面详解介绍具体的安装过程。

【实践案例 10-1】

从 MySQL 官方网站下载 Windows 版本的 MySQL 安装程序。例如，这里以 5.5.8 为例，得到名为 mysql-5.5.8-win32.msi 的文件。双击该文件开始安装。

（1）打开如图 10-1 所示的欢迎安装 MySQL 界面。

（2）单击 Next 按钮，弹出是否同意安装协议界面，如图 10-2 所示。在这里启用 I accept the terms in the License Agreement 复选框。



图 10-1　欢迎安装 MySQL 界面

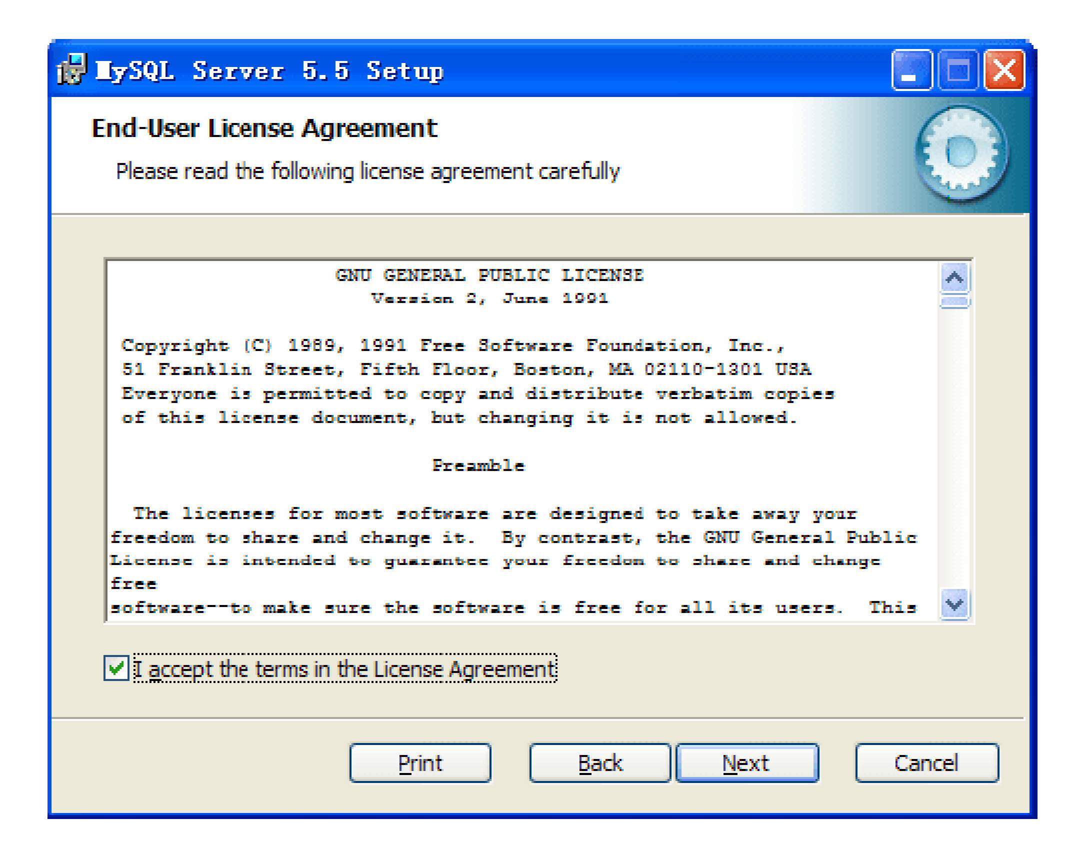


图 10-2　是否同意安装协议界面

（3）单击 Next 按钮，弹出选择安装类型界面，有 3 个选项：Typical（典型安装）、Custom（自定义安装）和 Complete（完全安装），如图 10-3 所示。

（4）单击 Custom 按钮，进入选择组件界面，如图 10-4 所示。

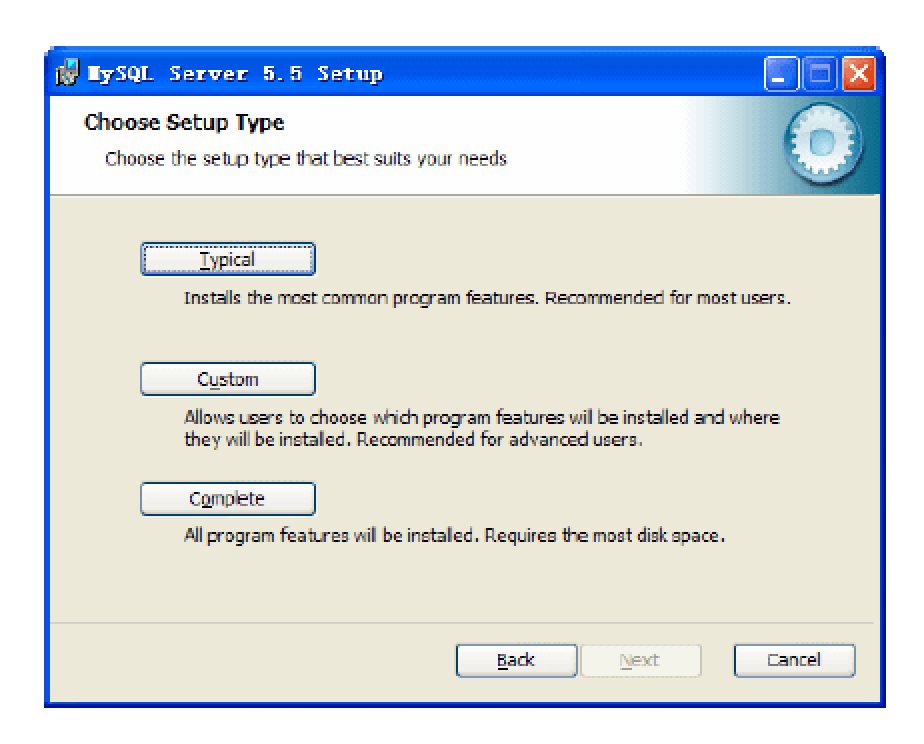


图 10-3　选择安装类型界面

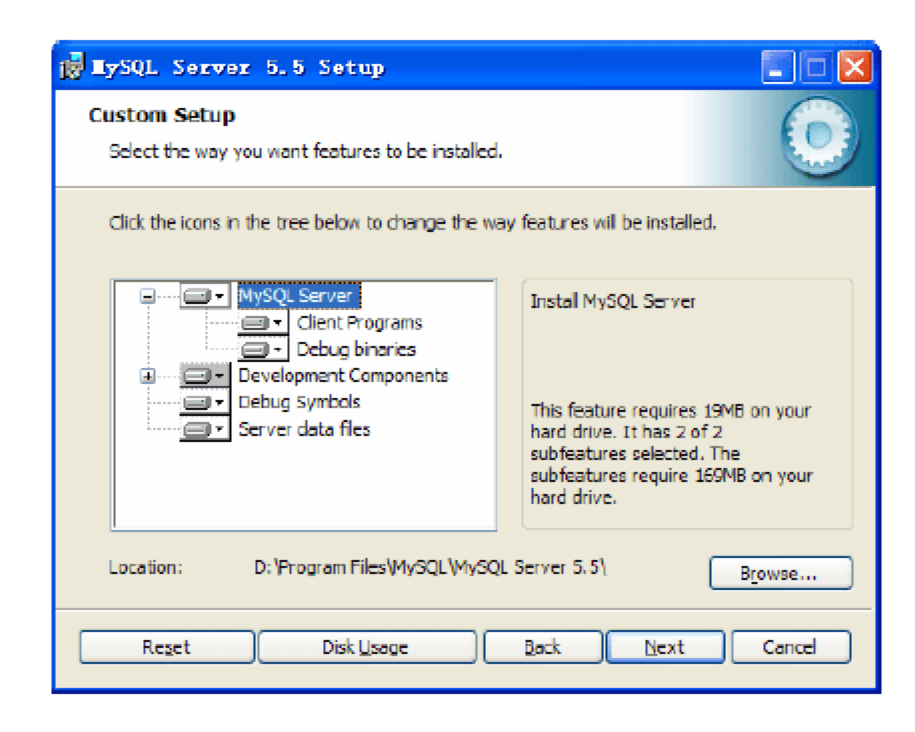


图 10-4　选择组件界面

（5）选择所有的组件之后单击 Browse 按钮改变安装路径，这里使用的安装路径为“D:\Program Files\MySQL\MySQL Server 5.5”。然后单击 Next 按钮，进入准备安装界面，如图 10-5 所示。

（6）单击 Install 按钮开始安装 MySQL，如图 10-6 所示。安装过程中，会弹出 MySQL Enterprise 界面，如图 10-7 所示。

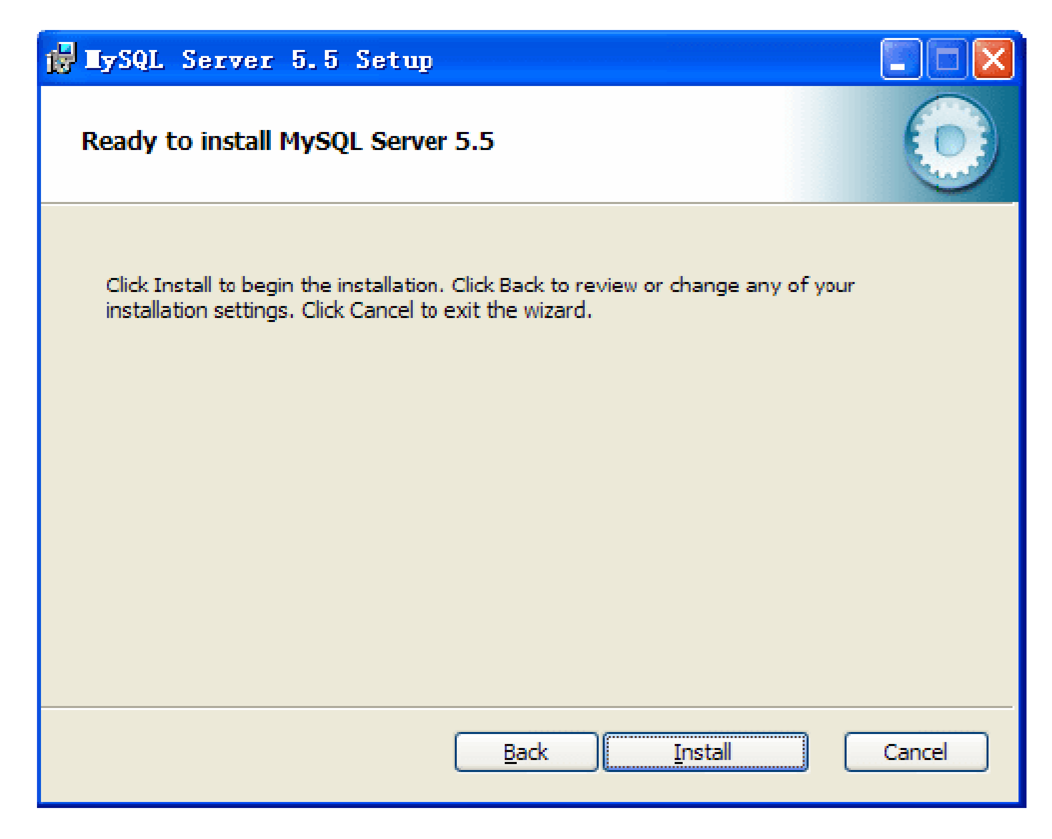


图 10-5　准备安装界面

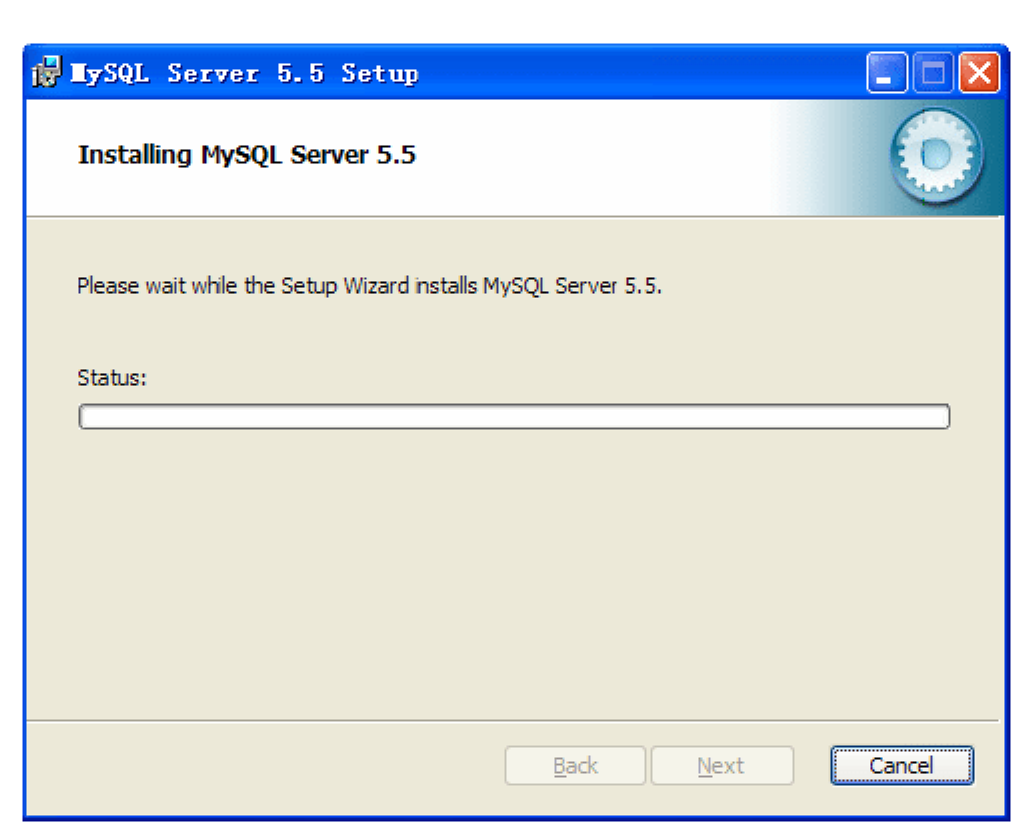


图 10-6　安装 MySQL 界面

择使用数据库类型的界面，如图 10-13 所示，在这里直接使用默认值。然后单击 Next 按钮，进入为数据库选择存储驱动器的界面，如图 10-14 所示。

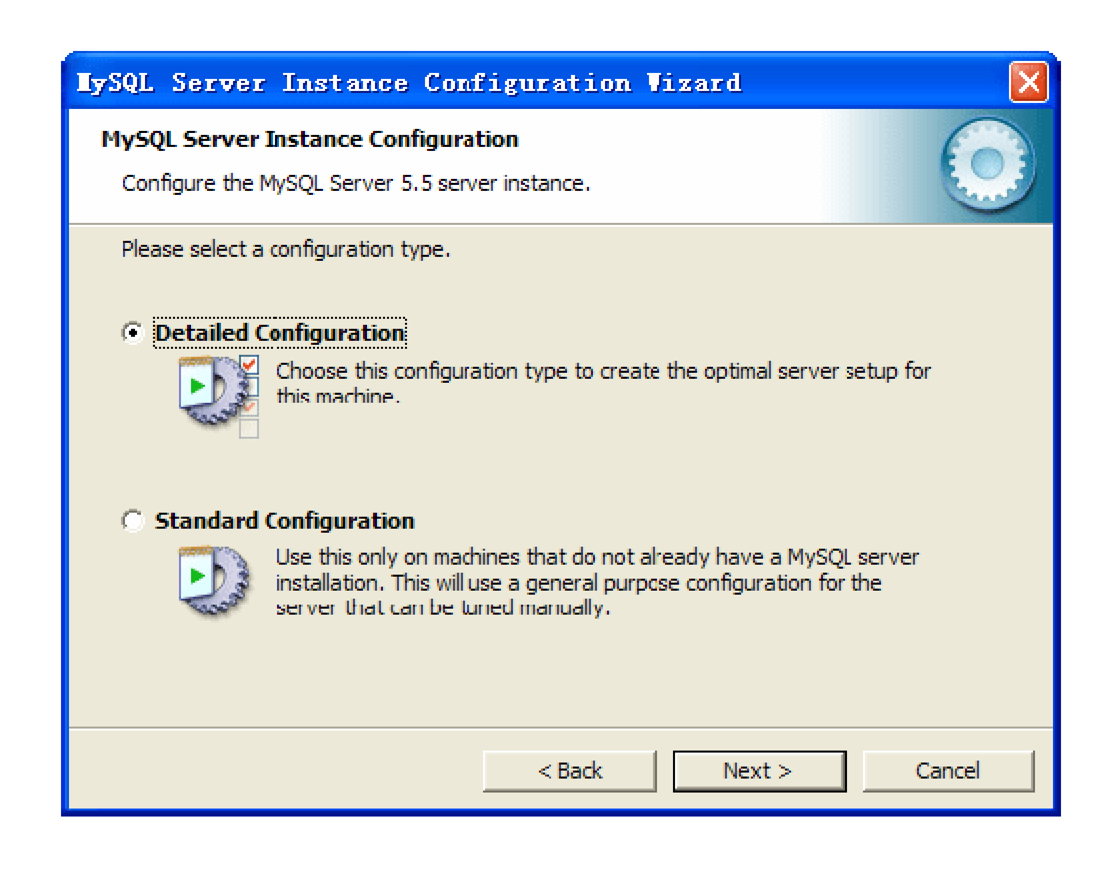


图 10-11　选择配置类型界面

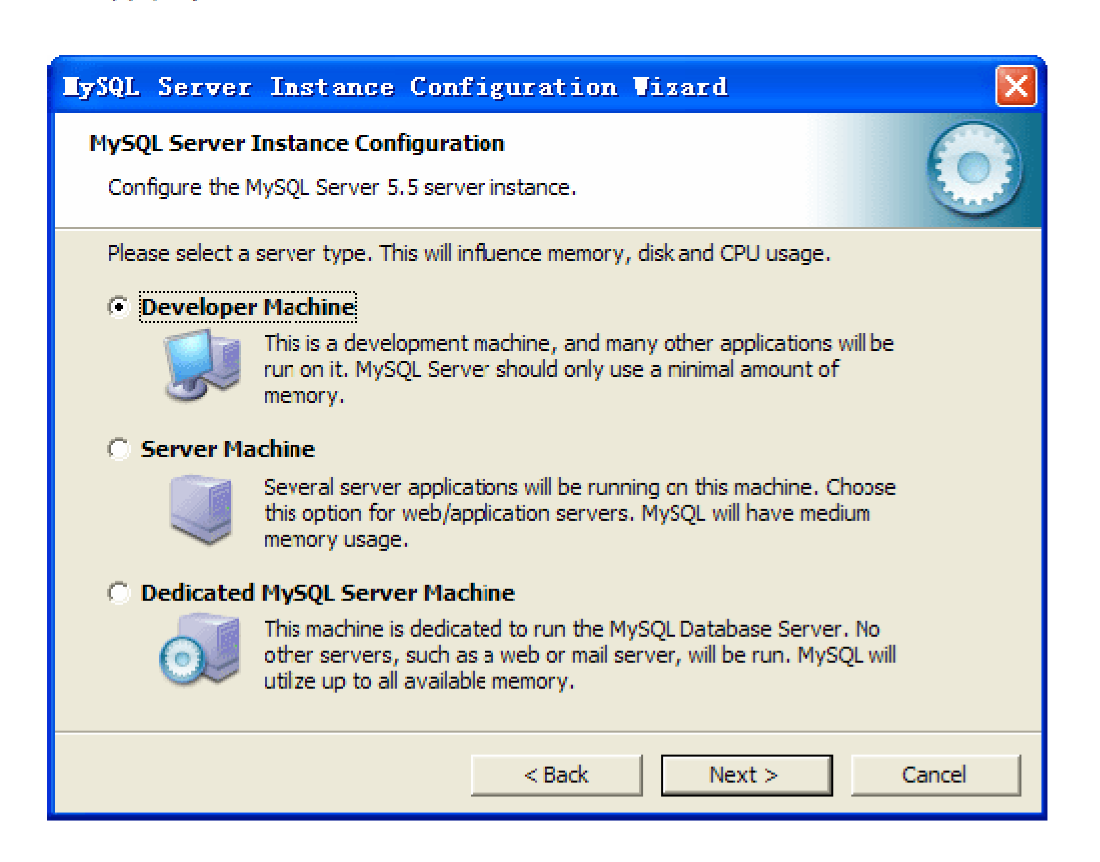


图 10-12　选择服务类型界面

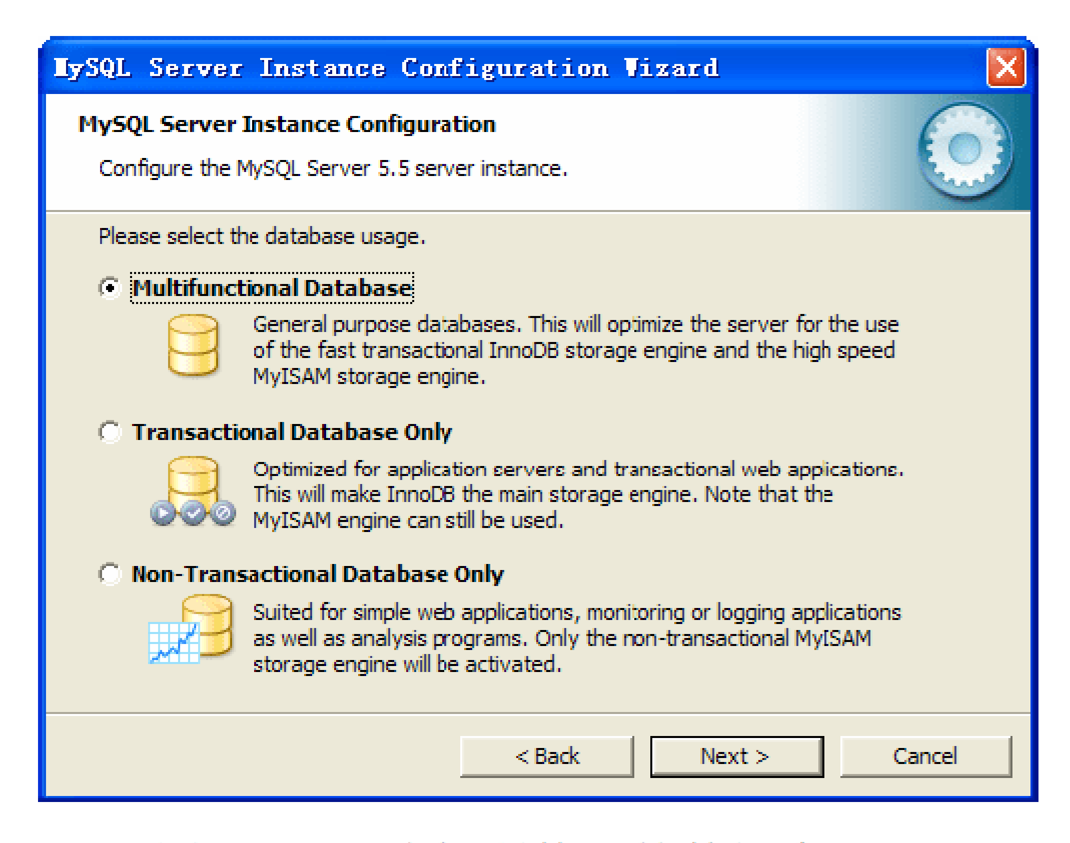


图 10-13　选择所使用的数据库

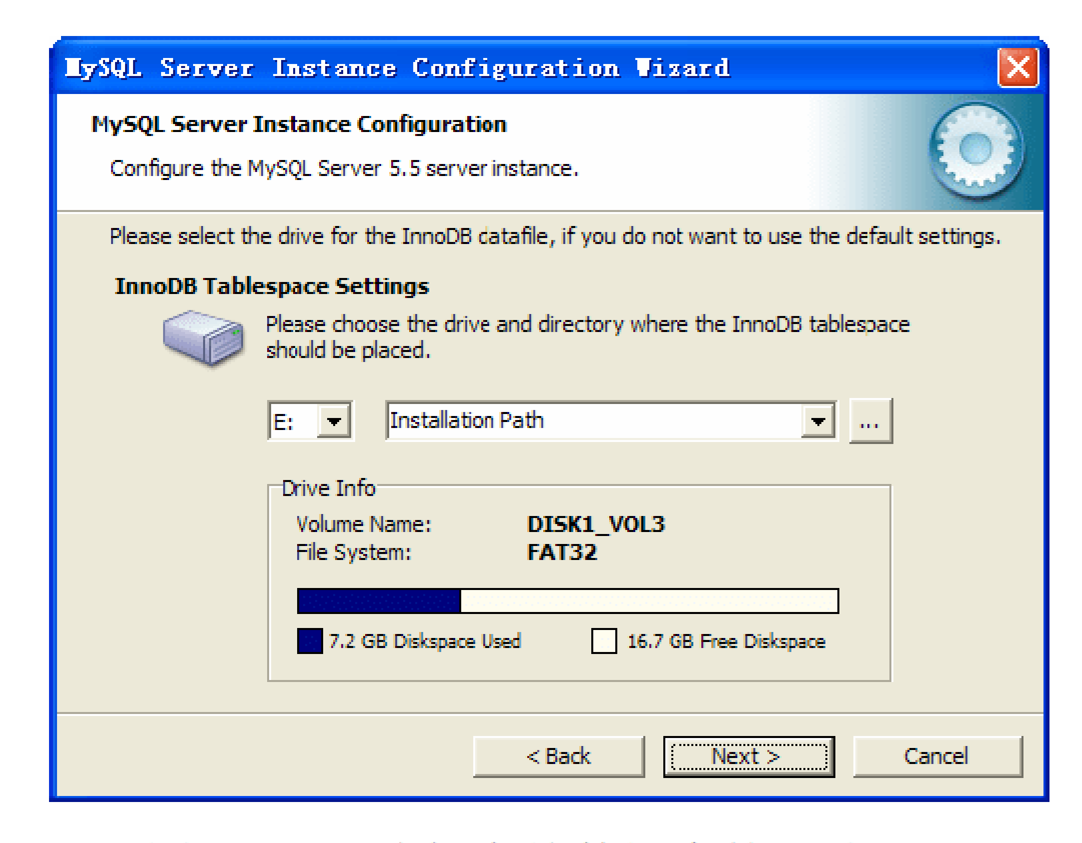


图 10-14　选择存储数据库的驱动器

（4）单击 Next 按钮，弹出选择同时最多执行的连接数的界面，如图 10-15 所示。这里同样选择默认值，即 Decision Support(DSS)/OLAP 选项。

（5）单击 Next 按钮，弹出选择数据库端口的界面，如图 10-16 所示。这里使用默认的 3306 端口号。

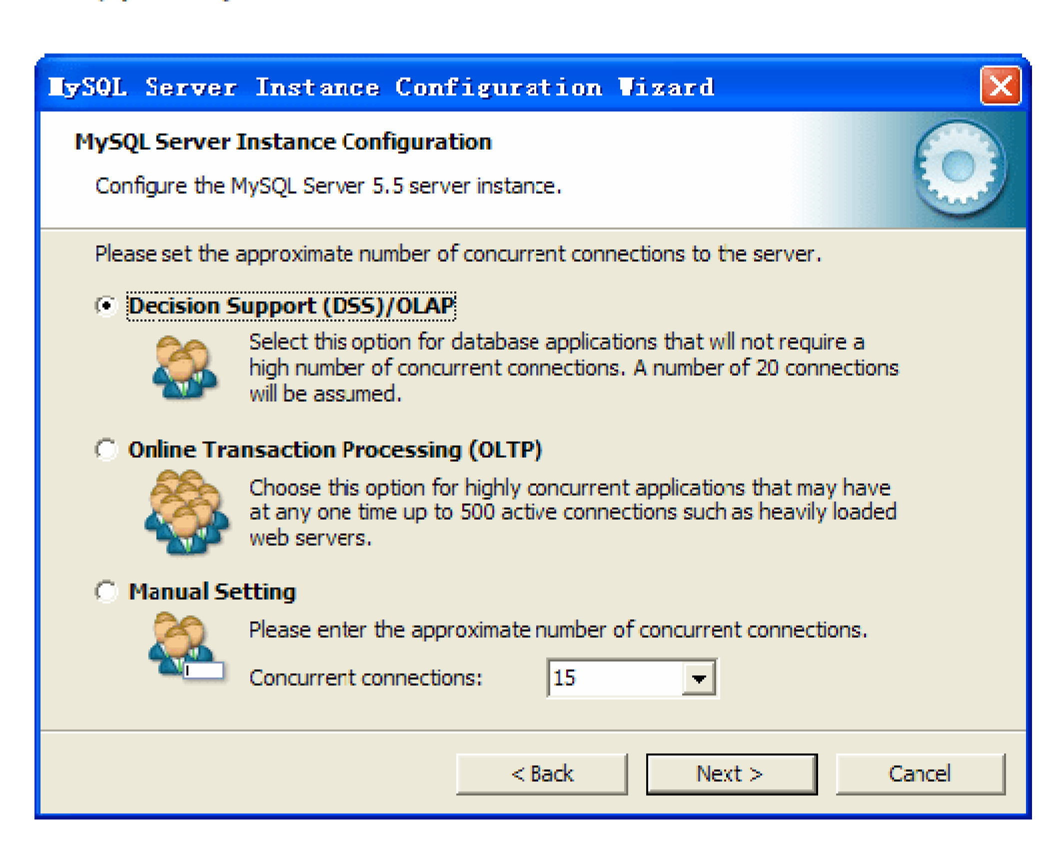


图 10-15　选择同时最多执行的线程数

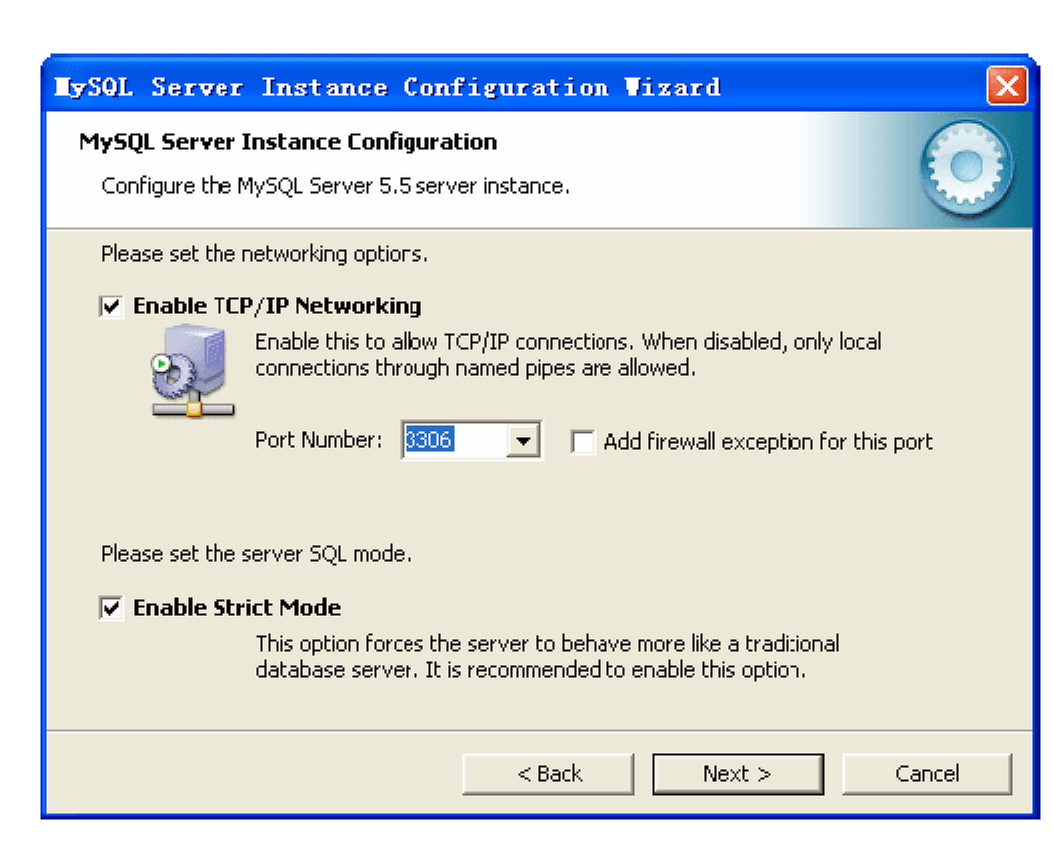


图 10-16　选择数据库端口号界面

（6）单击 Next 按钮，进入字符集选择界面，如图 10-17 所示。在这里选择最后一个选项，并且选择 utf8 字符集。



如果使用非 utf8 字符集，那么本书后面读取数据库中的数据时汉字将显示为乱码。

（7）选择字符集后，单击 Next 按钮，弹出配置 Windows 选项界面，如图 10-18 所示。在这里启用 Include Bin Directory in Windows PATH 复选框，该选项表示将 MySQL 路径添加到 Windows 的环境变量中。

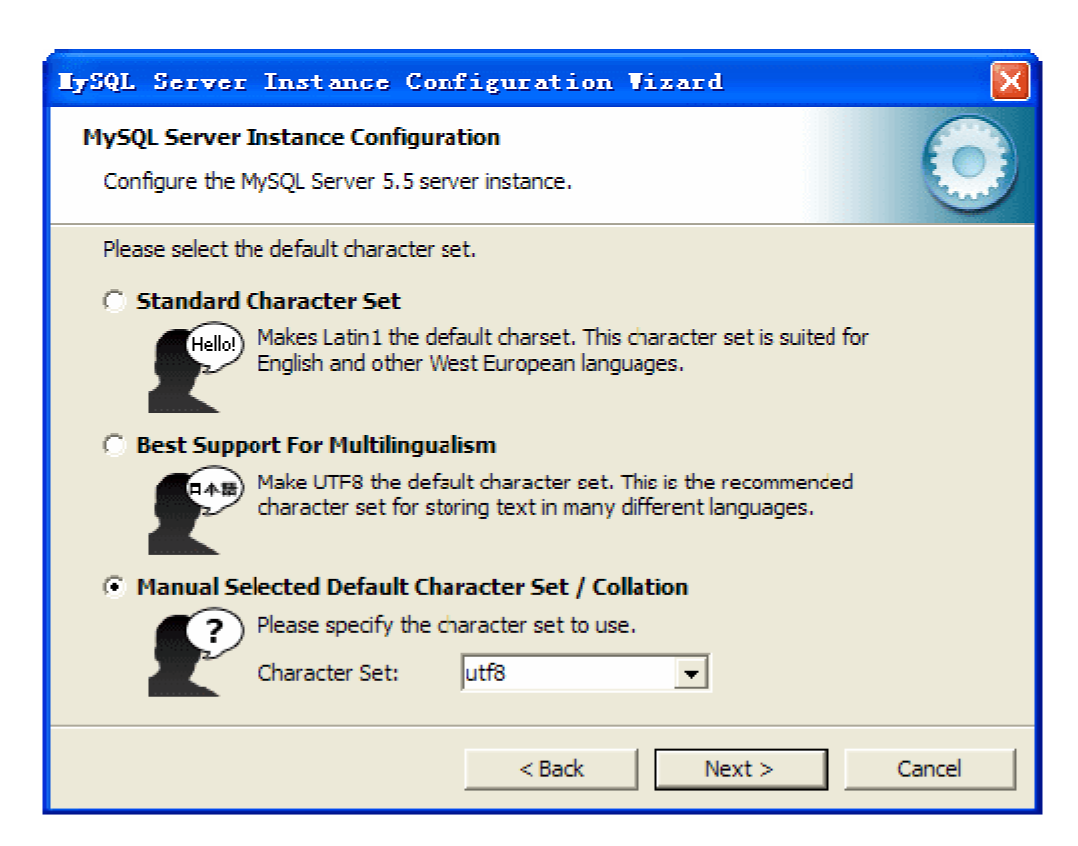


图 10-17　选择数据库使用的字符集



图 10-18　配置 Windows 选项

（8）单击 Next 按钮为 MySQL 数据库设置密码，如图 10-19 所示。设置完密码之后，单击 Next 按钮，弹出保存 MySQL 设置界面，如图 10-20 所示。

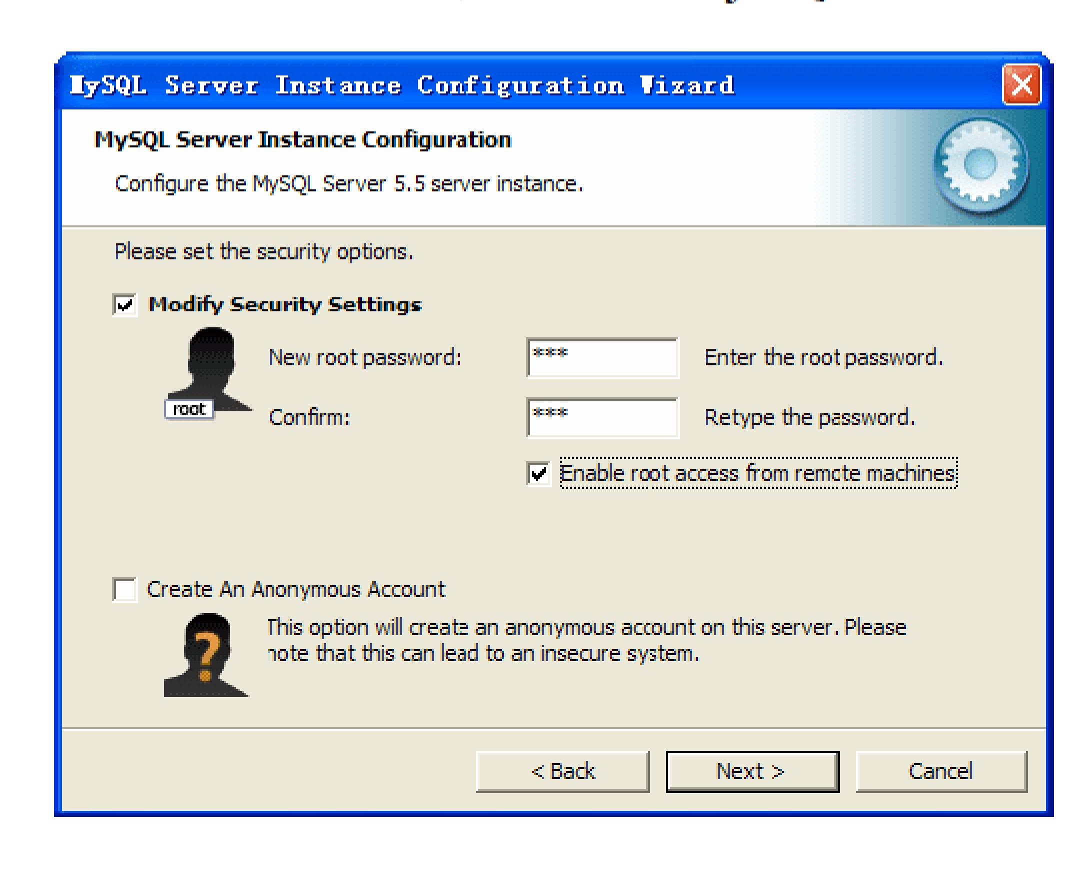


图 10-19　为数据库设置密码

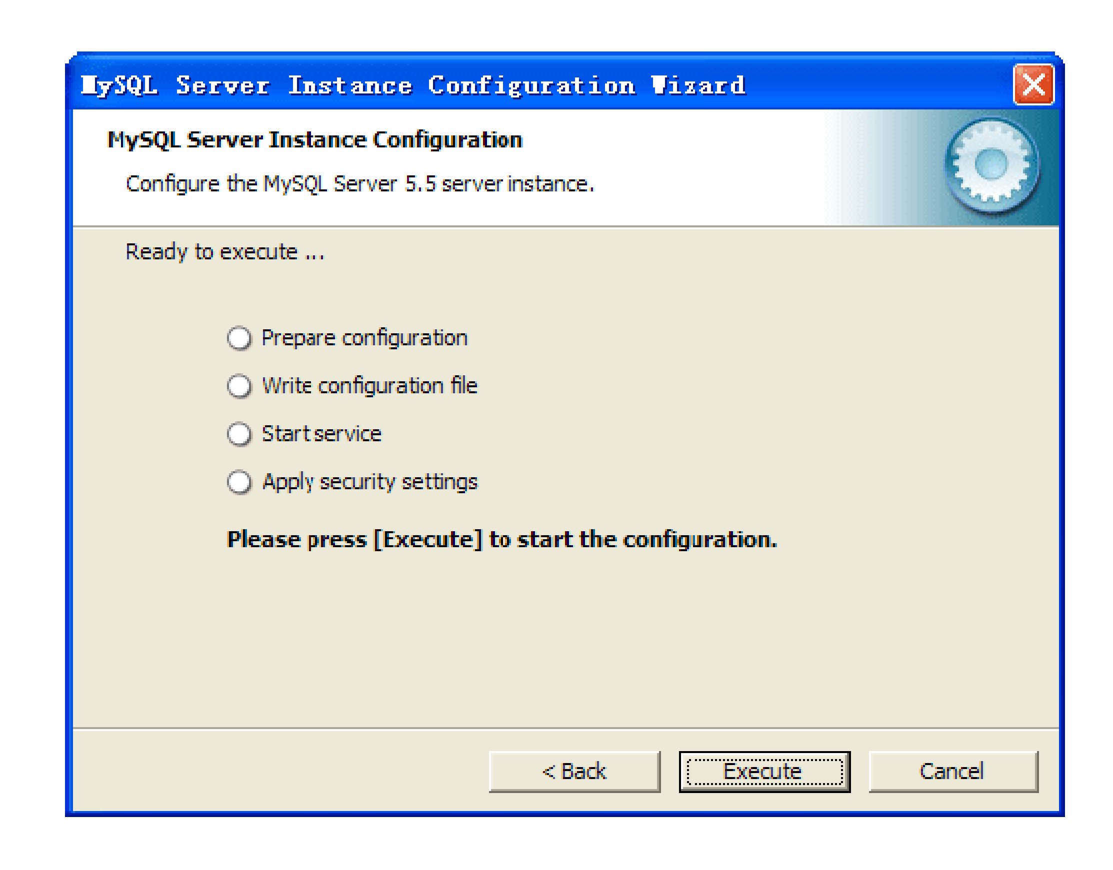
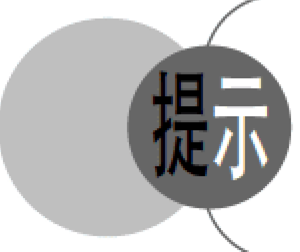


图 10-20　保存配置界面

（9）单击 Execute 按钮进行保存。如果保存成功，会弹出保存成功的界面，最后单击 Finish 按钮，完成配置。



手动配置 MySQL 数据库的方法是运行 MySQL 安装目录\bin\MySQLInstanceConfig.exe。

10.1.3 基本操作

本节主要介绍配置 MySQL 数据库完成之后使用 MySQL 数据库的内容。

1. 设置管理员密码

MySQL 数据库的系统管理员账号是 root，在配置时可以为它指定一个密码。如果要更改这些密码，首先应以 root 身份连接到 MySQL 服务器，然后通过 mysql -u root 命令完成。该命令使用 MySQL 客户程序连接到服务器。

出现 MySQL 提示符之后，使用如下命令为系统管理员设置密码：

```
mysql> set password for root=password("123456")
```

执行后，root 用户的密码设置为 123456。还可以使用同样的命令对其他用户进行设置或更改密码。

2. 退出 MySQL 服务器

在 MySQL>提示符下输入 quit 即可退出交互的操作界面。

```
[root@zht ~]# mysql -p
Enter password:
mysql> quit
Bye
```

或

```
mysql> \q
Bye
```

3. 查询操作

在 MySQL>提示符下直接输入 select 语句即可完成简单的查询操作。例如，下面的命令显示了 MySQL 服务器的版本号和当前日期。这里要注意的是，MySQL 命令不区分大小写。

```
mysql> select version(),current date();
+-----------+---------------------+
| version() | current date()    |
+-----------+---------------------+
| 5.5.8    | 2012-05-23      |
+-----------+---------------------+
1 row in set (0.22 sec)
```

4. 多行语句操作

如果一条 SQL 语句比较长，可将其分成多行来输入，并在最后输入分号“；”作为结束。

7. 打开数据库并显示

当 MySQL 数据库服务器上存在多个数据库时，可使用“use 数据库名”命令进入指定的数据库。使用“select database()”命令可显示当前所在的数据库。如下所示代码，在输入 use 和 quit 命令时不需要分号结束：

```
mysql> use mysql
Database changed
mysql> select database();
+----------------+
| database()   |
+----------------+
| mysql       |
+----------------+
1 row in set (0.00 sec)
```

8. 查看数据表

如果已经打开数据库，要查看其中包括哪些数据表，可使用如下命令：

```
mysql> show tables;
+----------------------------------------+
| Tables in mysql            |
+----------------------------------------+
| columns_priv              |
| db                      |
| func                    |
......
+----------------------------------------+
17 rows in set (0.00 sec)
```

9. 显示表内容

这里以显示 db 表的内容为例，可使用如下命令：

```
mysql> select * from db;
```

10.2 PHP 连接 MySQL 方式

前面学习了 MySQL 数据库的安装、配置和简单操作，本节主要介绍 PHP 与 MySQL 进行交互的两个库：mysql 和 mysqli。本节将对这两个库进行简单介绍，后面详细介绍具体的操作。

用用户 root 和密码 root 连接本地的 MySQL 服务器，将连接保存到$link 变量中。

```
//localhost 表示本地的 MySQL 服务器
$link=mysql_connect("localhost","root","root");
```

mysql_connect()除了上面的连接方式外，还可以使用下面几种形式：

```
$link = mysql_connect("db.example.com:3721", "root", "password");
                                           //使用域名
$link = mysql_connect("192.162.10.6:3721", "root", "password");
                                           //使用 IP 地址
$link = mysql_connect("/usr/local/tmp/mysql", "root", "password");
                                           //使用路径
$link = mysql_connect("localhost:/usr/local/tmp/mysql.sock", "root",
"password");                               //使用文件
```

【实践案例 10-2】

使用 mysql_connect()函数测试到 MySQL 数据库的连接是否成功，并给出相应的结果。实现代码如下所示：

```
<?php
    $host="localhost";              //MySQL 服务器地址，localhost 表示本机
    $user="root";                   //登录用户
    $password="123456";             //登录密码
    $con = mysql_connect($host,$user,$password);    //建立连接
    if ($con)                       //判断连接结果
    {                               //输出成功信息
        echo "数据库连接成功。";
    }
    else
    {                               //输出错误信息
        echo "数据库连接失败。<br/>".mysql_error();
    }
?>
```

在上述代码中，调用 mysql_connect()函数进行数据库连接，并在函数中提供了 3 个参数，第一个$host 表示要连接的 MySQL 服务器，这里值为 localhost 表示本机；$user 和$password 分别表示登录用户名和密码。3 个参数合到一起表示，使用用户名 root 和密码 123456 连接到本地（localhost）的 MySQL 数据库。

在$con 变量中保存了 mysql_connect()函数的返回值。如果连接成功，则执行 if 语句中的代码，否则执行 else 语句。

将代码保存为 mysql_connect.php 再运行。如果连接失败，则会提示连接失败的原因，如图 10-22 所示。否则在页面中输出字符串“数据库连接成功”，如图 10-23 所示。



如果出现的信息是“Fatal error: Call to undefined function mysql_connect()”，则表示 mysqli 库没有被正确的引入。

使用 mysql_connect()函数建立的连接在脚本执行结束后，连接也就关闭了。为此 mysql 库提供了 mysql_pconnect()函数可以建立一个持久化连接。该函数的语法格式如下所示：



图 10-22　连接失败

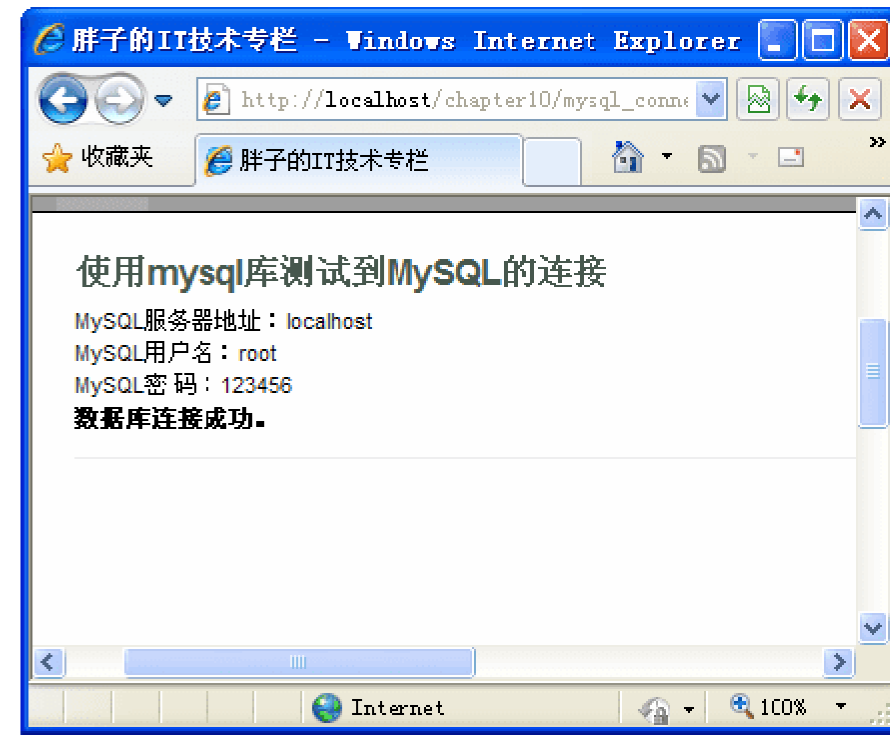


图 10-23　连接成功

```
resource mysql_pconnect ([ string $server [, string $username [, string
$password [, int $client_flags ]]]] )
```

各个参数含义与 mysql_connect()函数相同，在此处就不介绍。下面的示例代码使用 mysql_pconnect()函数建立一个到 MySQL 数据库，并给出相应的结果：

```
<?php
    $con = mysql pconnect("localhost","root","123456");
                                            //建立持久化连接
    if ($con) {                             //判断连接结果
        echo "恭喜您！数据库连接成功。";    //输出成功信息
    }
    else{                                   //输出错误信息
        echo "错误！数据库连接失败:".mysql_error();
    }
?>
```

在上述语句中，当脚本执行完毕后到 MySQL 服务器的连接$link 不会被关闭。此连接将保持打开状态以备下次使用。即使使用 mysql_close()函数也不会关闭由 mysql_pconnect()函数建立的持久连接。

mysql_pconnect()和 mysql_connect()非常相似，都可以建立到 MySQL 的连接，它们主要有如下区别。

- **当连接的时候 mysql_pconnect()函数将先尝试寻找一个在同一个主机上用同样的用户名和密码已经打开的（持久）连接，如果找到，则返回此连接标识而不打开新连接。**
- **当脚本执行完毕后到 MySQL 服务器的连接不会被关闭，此连接将保持打开以备以后使用。**

```
        echo "数据库连接成功。<br/>";
        $db_selected = mysql_select_db("db_blog", $link);
                                                      //选择 db_blog 数据库
        if($db_selected) {                            //选择数据库成功
            echo "已经成功选择 db_blog 数据库。<br/>";
            //对 db_blog 数据库的操作
        }
        else {                                        //选择数据库失败
            echo " db_blog 数据库选择失败。<br/>";
        }
        $close=mysql_close($link);                    //关闭连接
        if($close) echo "数据库已经关闭。";
    }
?>
```

在上述代码中，db_blog 表示在 MySQL 服务器中创建好的数据库。当选择了数据库后，便可以对 db_blog 数据库和其中的表进行各种操作，最后调用 mysql_close()函数关闭当前的连接。

10.4　基本操作

经过前面的学习，读者已经掌握了操作 MySQL 数据库的第一步。接下来则需要从数据库中获取所需的数据并让它们显示到页面上，或者执行一条 SQL 语句更新数据。

本节主要讲解在 PHP 中如何获取和显示数据，以及执行 SQL 语句。

10.4.1　获取结果集

当在 MySQL 中执行一条查询语句时，MySQL 会自动将语句发送到指定的数据库，并将获取执行的结果，然后以行和列的形式展现出来，同时输出总行数。

如果要使用 PHP 获取 MySQL 查询的结果集必须依赖于函数。下面将介绍通过函数实现获取查询结果的记录数、获取其中的一列或者一行等。

1. 获取查询结果的记录数

mysql_num_rows()函数用于获取 select 语句查询得到数据集的行数，语法格式如下所示：

```
int mysql_num_rows ( resource $result )
```

其中，参数$result 为 mysql_query()函数所返回的结果集，执行后返回结果集中的记录数。

【实践案例 10-4】

假设在 MySQL 的 db_blog 数据库中有一个 articles 表，其中保存了有关博客文章的信息，像 id（文章编号）、title（标题）、a_id（作者编号）、views（浏览量）、c_id（类别编号）和 property（属性）等。

现在要从 articles 表中查询出如下信息。

- **查询总共有多少文章。**
- **查询浏览量大于 5 的文章数量。**
- **查询类别编号为 3 的文章数量。**

要实现上述的查询功能，需要经过连接 MySQL、选择 db_blog 数据库、执行 SQL 查询语句、获取结果、关闭连接这样几步。最终的实现代码如下所示：

```
<?php
$conn=mysql connect("localhost","root","123456");
if($conn)
{
    echo "(1)连接 MySQL 成功。<br/>";
    if(mysql select db("db blog",$conn)){
        echo "(2)选择 db_blog 数据库成功。<br/>";
    }
    echo "(3)查询结果如下。<br/>"
?>
<ul>
<?php
    $strsql="select id from articles";                //第一个查询语句
    $result=mysql query($strsql,$conn);               //执行查询
    $rows=mysql_num_rows($result);                    //获取查询的结果数
    echo "<li>当前共有文章: $rows 条</li>";           //输出
    $strsql="select id from articles where views>5";
    $result=mysql_query($strsql);                     //执行第二个查询
    $rows=mysql num rows($result);
    echo "<li>浏览量在 5 以上的有: $rows 条</li>";
    $strsql="select id from articles where c id=3";
    $result=mysql_query($strsql);                     //执行第三个查询
    $rows=mysql num rows($result);
    echo "<li>类别 3 下的文章有: $rows 条</li>";
?>
</ul>
<?php
    if(mysql_close($conn)) echo "(4)数据库已经关闭。";    //关闭连接
}
?>
```

如上述代码所示，在实现时用到的函数有 mysql_connect()、mysql_select_db()、mysql_query()、mysql_num_rows()和 mysql_close()。将上述代码保存为 mysql_num_rows.php，在浏览器中执行，效果如图 10-24 所示。

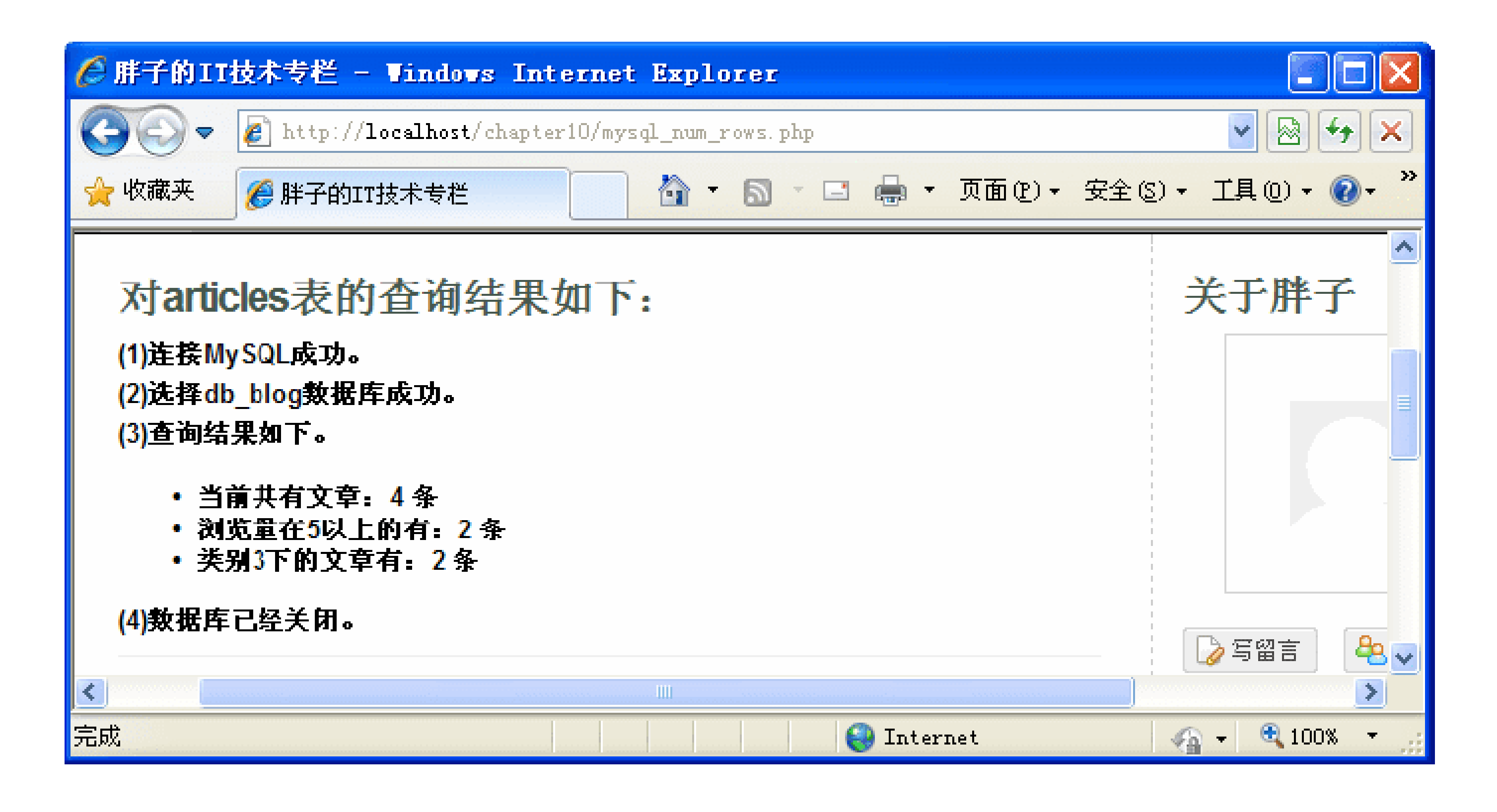


图 10-24　查询结果



如果执行的是 insert、update 或 delete 语句，可以用 mysql_affected_rows()函数来获取数据访问时所影响的记录行数，该函数没有参数。

2．获取结果集中的一列

mysql_result()函数可以返回结果集中的一个列，该函数语法格式如下所示：

```
mixed mysql_result ( resource $result , int $row [, mixed $field ] )
```

该函数可以从资源对象$result 中指定的$row 中获取一个指定的$field 数据。其中，$row 参数可以是列的偏移量，$field 表示要获取的列名（或者别名）。如果 mysql_result()函数执行成功返回列值，否则返回 false。

【实践案例 10-5】

在 db_blog 数据库的 articles 表中按 id 排序，然后输出第 1 篇文章的编号、标题、发布时间、浏览量和标签。实现代码如下所示：

```
<h2>显示文章信息</h2>
<ul>
<?php
$conn=mysql_connect("localhost","root","123456");
if($conn)
{
    mysql_select_db("db_blog",$conn);
    $strsql="select id,title,pubtime,views,tags from articles order by id";
        //查询语句
    $result=mysql_query($strsql,$conn);          //执行查询
    $rows=mysql_num_rows($result);               //获取查询的结果数
```

```
    if($rows>0){                                    //判断是否大于 0
        for($index=0;$index<$rows;$index++){
            echo "<tr>";
            echo "<td>".mysql result($result,$index,"id")."</td>";
                                                    //输出结果
            echo "<td>".mysql result($result,$index,"title")."</td>";
            echo "<td>".mysql_result($result,$index,"pubtime")."</td>";
            echo "<td>".mysql_result($result,$index,"views")."</td>";
            echo "<td>".mysql result($result,$index,"tags")."</td>";
            echo "</tr>";
        }
    }
    mysql close($conn);                             //关闭连接
}
?></table>
```

上述代码使用循环对结果集进行遍历，然后将索引传递到 mysql_result()函数中进行获取，最后再输出。再次运行将看到如图 10-26 所示的效果。

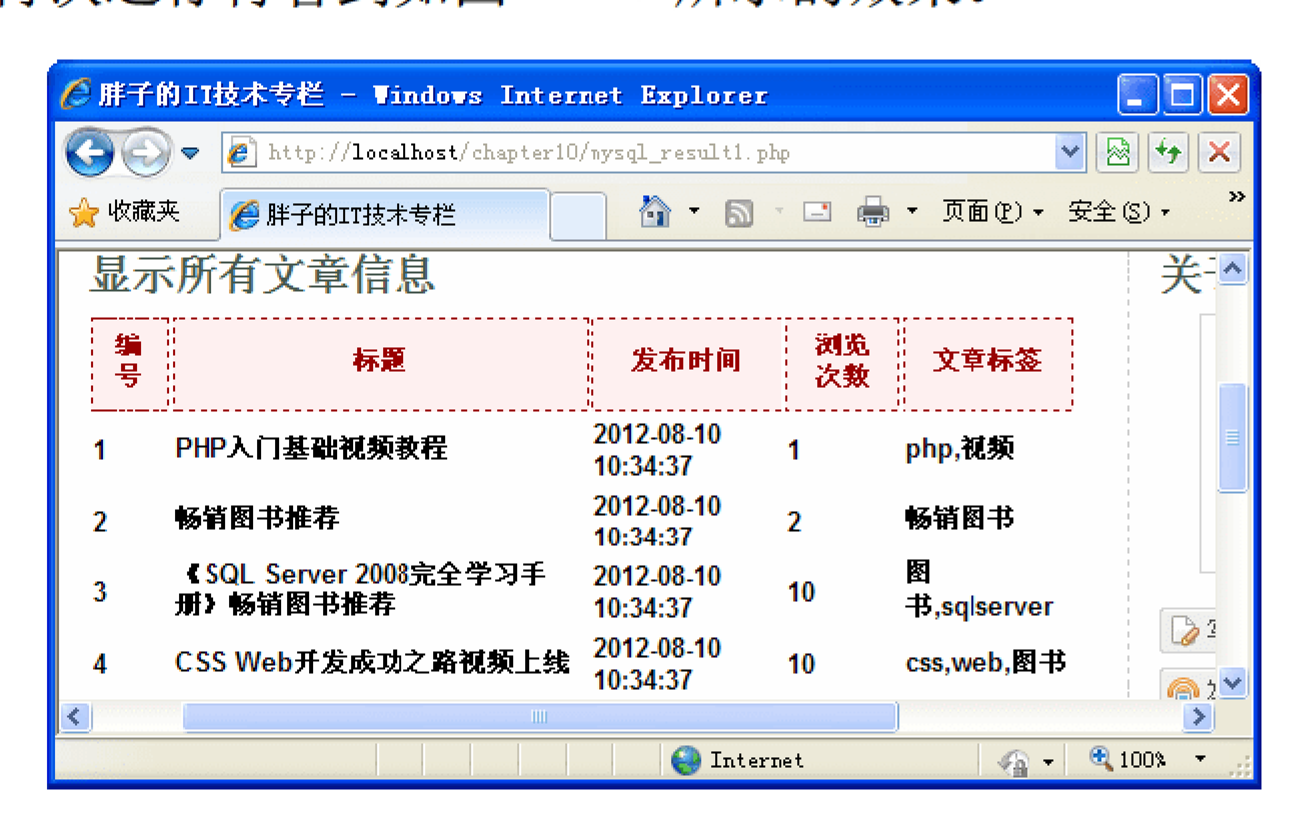


图 10-26　显示所有文章信息

3．获取数据集中的一行

mysql_fetch_row()函数可以从结果集中取得一行作为数组，该函数的语法格式如下所示：

```
array mysql_fetch_row ( resource $result )
```

mysql_fetch_row()函数从$result 关联的结果集中取得一行数据并作为数组返回。每个结果的列储存在一个数组的单元中，偏移量从 0 开始。

【实践案例 10-6】

在使用 mysql_result()函数遍历结果集时需要获取总行数，然后使用 for 语句来实现。使用 mysql_fetch_row()函数也可以遍历结果集，而且可以直接获取其中的一行。

例如下面的代码实现了遍历 articles 表中的所有文章数据：

```
<table>
<tr>
<th>编号</th><th>标题</th><th>发布时间</th><th>浏览次数</th><th>文章标签</th>
</tr>
<?php
$conn=mysql connect("localhost","root","123456");
if($conn)
{
    mysql select db("db blog",$conn);
    $strsql="select id,title,pubtime,views,tags from articles order by id";
    //查询语句
    $result=mysql_query($strsql,$conn);          //执行查询
    while(list($id,$title,$pubtime,$views,$tags)=mysql fetch row($result)){
                                                  //遍历结果集
        echo "<tr>";
        echo "<td>".$id."</td>";                  //输出 id 列
        echo "<td>".$title."</td>";
        echo "<td>".$pubtime."</td>";
        echo "<td>".$views."</td>";
        echo "<td>".$tags."</td>";
        echo "</tr>";
    }
    mysql_close($conn);                       //关闭连接
}
?></table>
```

在上述代码中，使用 while 语句循环调用 mysql_fetch_row()获取结果集中的一行，如果没有行时则返回 false 退出。运行效果与使用 mysql_result()函数进行遍历相同。

10.4.2 显示结果集

PHP 提供了一些函数用于显示在数据库中执行查询得到的数据集，分别是：mysql_fetch_assoc()函数、mysql_fetch_array()函数和 mysql_fetch_object()函数。

1. mysql_fetch_assoc()函数

mysql_fetch_assoc()函数可以从结果集中取得一行作为关联数组并返回，如果没有更多的行，则返回 false。语法格式如下所示：

```
array mysql_fetch_assoc ( resource $result )
```

【实践案例 10-7】

下面的代码将 articles 表所有数据保存到结果集，然后输出对结果集调用 mysql_fetch_assoc()函数后返回的关联数组。

```
<h2>使用 mysql_fetch_assoc()获取文章信息</h2>
```

```
<?php
$conn=mysql_connect("localhost","root","123456");
if($conn)
{
    mysql_select_db("db_blog",$conn);
    $strsql="select id,title,pubtime,views,tags from articles order by id";
                                                    //查询语句
    $result=mysql_query($strsql,$conn);             //执行查询
    echo "<pre>";
    print_r(mysql_fetch_assoc($result));
                        //输出调用 mysql_fetch_assoc()函数返回的关联数组
    echo "</pre>";
    mysql_close($conn);                             //关闭连接
}
?>
```

上述代码运行后会获取结果集中的第 1 行并输出，效果如图 10-27 所示。

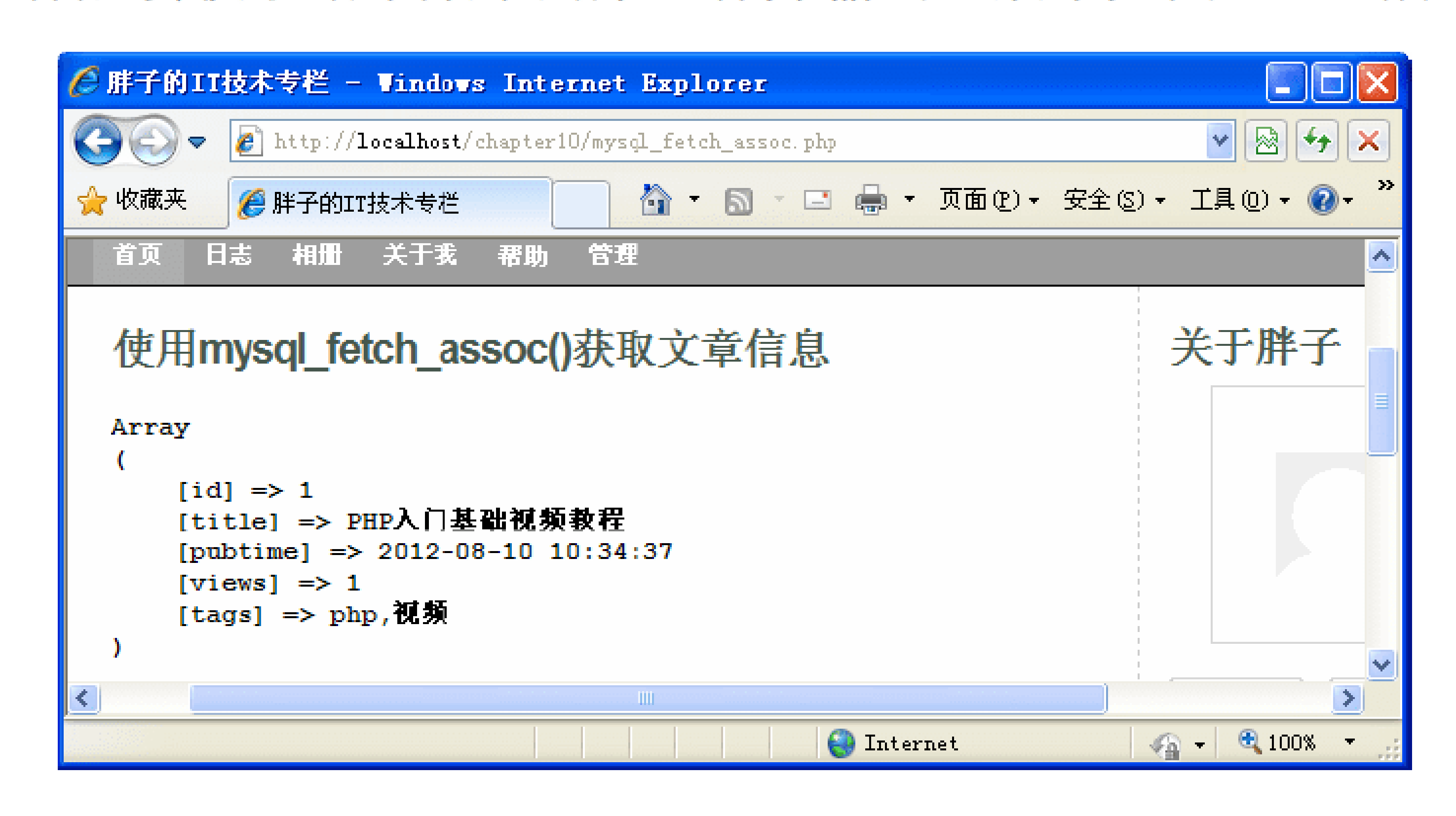


图 10-27　输出关联数组

例如，同样是遍历 articles 表的数据，使用 mysql_fetch_assoc()函数的实现代码如下所示：

```
<table>
<tr>
<th>编号</th><th>标题</th><th>发布时间</th><th>浏览次数</th><th>文章标签</th>
</tr>
<?php
$conn=mysql_connect("localhost","root","123456");
if($conn)
{
    mysql_select_db("db_blog",$conn);
    $strsql="select id,title,pubtime,views,tags from articles order by id";
                                                        //查询语句
    $result=mysql_query($strsql,$conn);                 //执行查询
    while($row= mysql_fetch_assoc($result)){
```

```
            echo "<tr>";
            echo "<td>".$row['id']."</td>";              //输出结果
            echo "<td>".$row['title']."</td>";
            echo "<td>".$row['pubtime']."</td>";
            echo "<td>".$row['views']."</td>";
            echo "<td>".$row['tags']."</td>";
            echo "</tr>";
        }
        mysql_close($conn);                              //关闭连接
    }
    ?></table>
```

如上述代码所示，mysql_fetch_assoc()函数从结果集$result 中取出一行作为数组保存到$row 变量中，然后使用 while 语句循环读取。取值的时候要注意，指定的数组键名必须与表中的列表一致，如果大小写不同将导致无法识别。

2. mysql_fetch_array()函数

mysql_fetch_array()函数是 mysql_fetch_row()函数的扩展版本。它除了将数据以数字索引方式储存在数组中之外，还可以将数据作为关联索引储存，用列名作为键名。语法格式如下所示：

```
array mysql_fetch_array ( resource $result [, int $ result_type ] )
```

这里的可选参数$result_type 用于设置返回数组的格式，它有如下 3 个可选值。

- **MYSQL_ASSOC　关联数组，效果与 mysql_fetch_assoc()函数相同。**
- **MYSQL_NUM　数字数组。**
- **MYSQL_BOTH　默认，同时产生关联和数字数组。**

【实践案例 10-8】

下面的代码将 articles 表所有数据保存到结果集，然后输出对结果集调用 mysql_fetch_array()函数后返回的数组。

```
<h2>使用 mysql_fetch_array()获取文章信息</h2>
<?php
$conn=mysql_connect("localhost","root","123456");
if($conn)
{
    mysql_select_db("db_blog",$conn);
    $strsql="select id,title,pubtime,views,tags from articles order by id";
    //查询语句
    $result=mysql_query($strsql,$conn);     //执行查询
    echo "<pre>";
    print_r(mysql_fetch_array($result));    //输出返回的关联数组和数字数组
    echo "</pre>";
    mysql_close($conn);                     //关闭连接
}
?>
```

上述代码调用 mysql_fetch_array()时指定了一个参数，此时会使用默认的 MYSQL_BOTH 同时输出包含有关联和数字的数组。执行效果如图 10-28 所示。

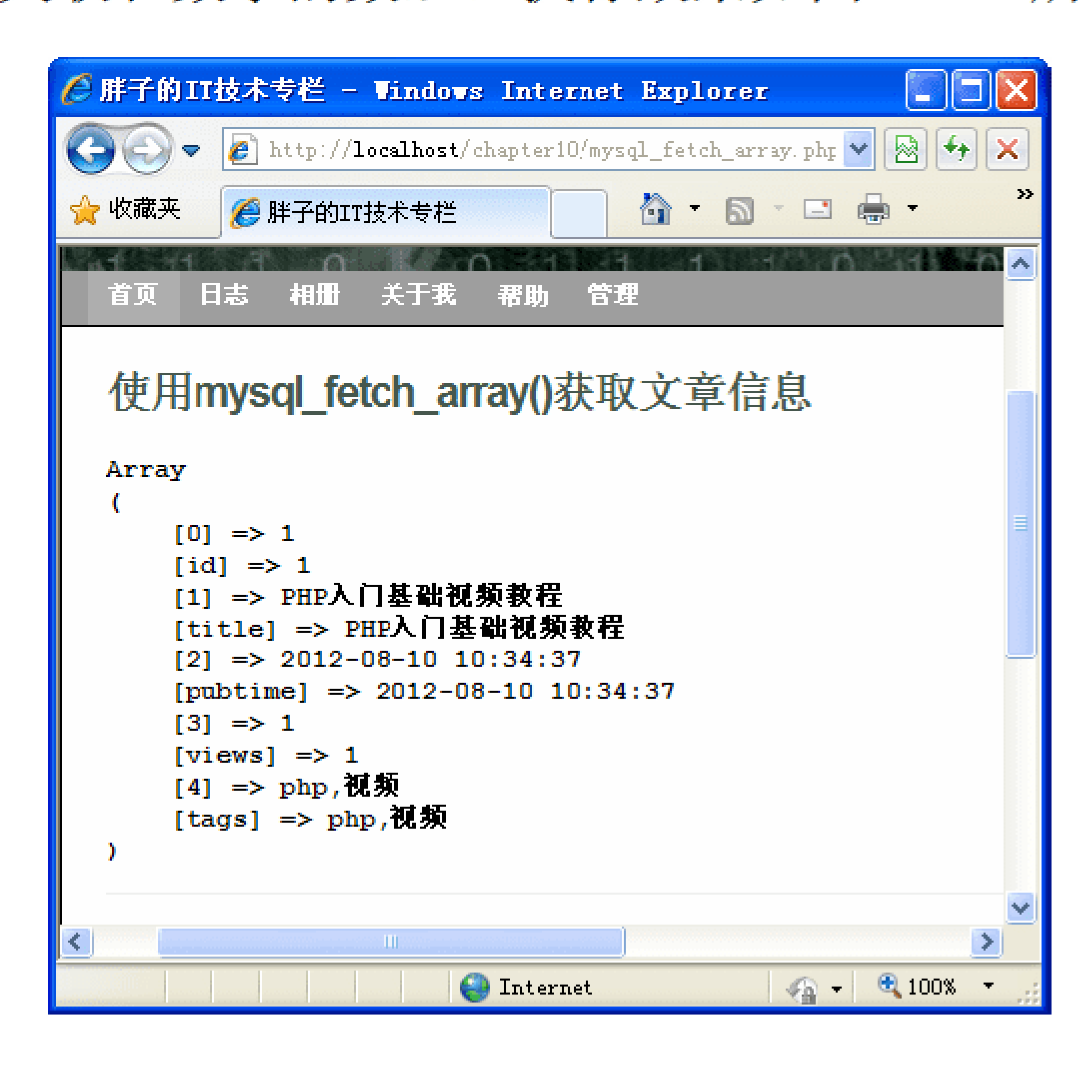


图 10-28　使用 mysql_fetch_array()函数

3．mysql_fetch_object()函数

mysql_fetch_object()函数从结果集中获取一行数据作为对象，该函数的语法格式如下所示：

```
object mysql_fetch_object ( resource $result )
```

mysql_fetch_object()函数的功能和 mysql_fetch_array()函数类似，唯一的区别是：该函数返回的是一个对象而不是数组。也就是说：只能通过列名来访问对象，而不是偏移量，并且列区分大小写。

下面代码使用 mysql_fetch_object()函数实现输出 articles 表的 id、title 和 views 列信息。

```
<h2>显示所有文章信息</h2>
<ul>
<?php
$conn=mysql_connect("localhost","root","123456");
if($conn)
{
    mysql_select_db("db_blog",$conn);
    $strsql="select id,title,views from articles order by id"; //查询语句
    $result=mysql_query($strsql,$conn);                        //执行查询
    while($row= mysql_fetch_object($result)){
        $id=$row->id;
        $title=$row->title;
        $pubtime=$row->views;
        echo "<li>编号: ".$id." <br/> 标题: ".$title." |  浏览次数:
        ".$pubtime."<br/><br/></li>";
    }
```

```
    mysql_close($conn);                                    //关闭连接
}
?></ul>
```

代码比较简单，这里就不再解释，浏览器中的执行效果如图 10-29 所示。

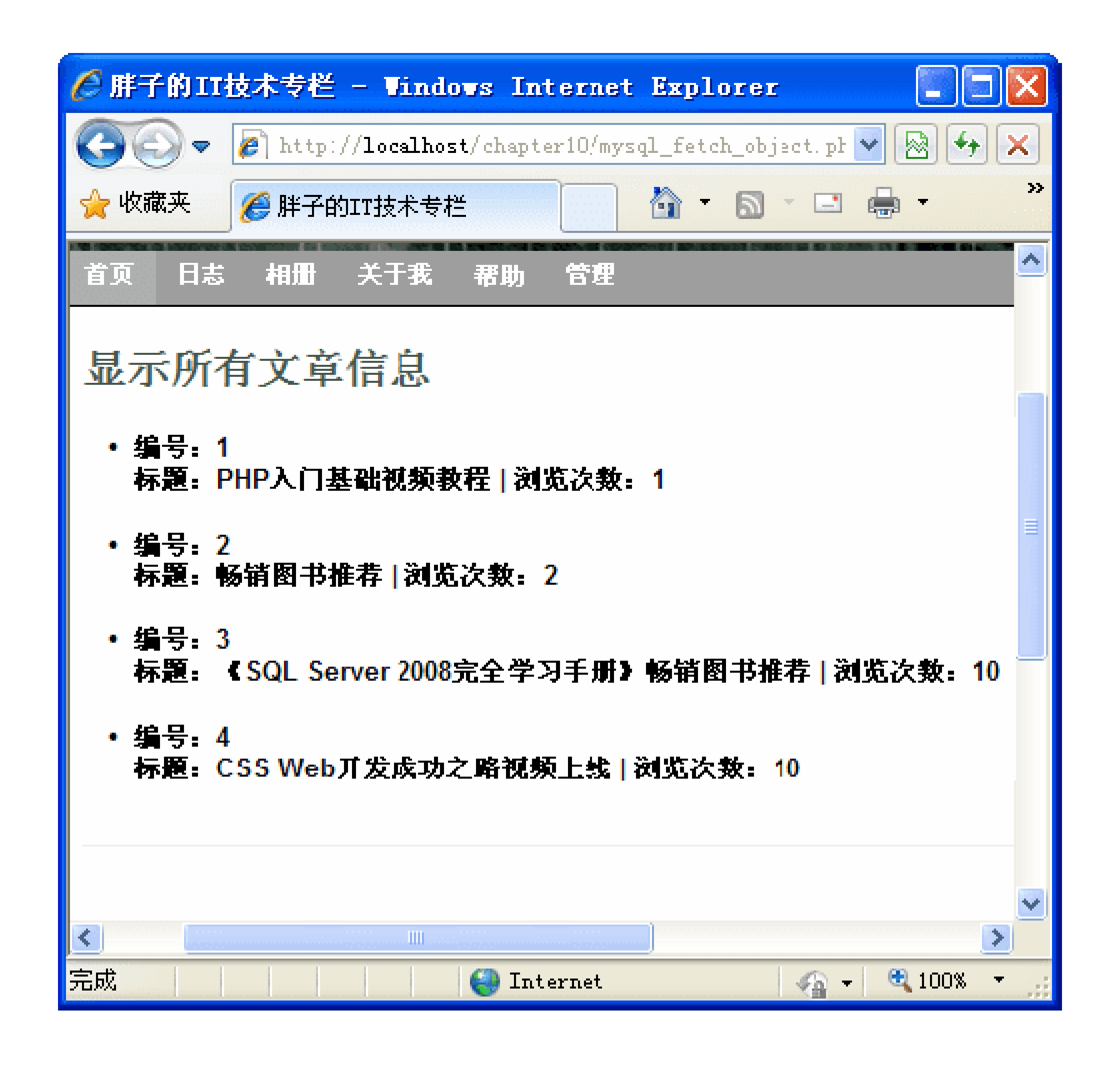


图 10-29　使用 mysql_fetch_object()函数

10.4.3　执行 SQL 语句

前面已经多次使用 mysql_query()函数向 MySQL 发送 select 语句，该函数还可以执行 update、delete 和其他数据库语句。

除了 mysql_query()函数还可以使用 mysql_db_query()函数把 SQL 语句传递给 MySQL，然后由 MySQL 执行这些 SQL 语句。

1. mysql_query()函数

mysql_query()函数用于向 MySQL 发送一条 SQL 语句，函数语法格式如下所示：

```
resource mysql_query ( string $query [, resource $link_identifier ] )
```

其中，$query 参数表示要执行的 SQL 语句， $link_identifier 参数表示打开的数据库连接。若没有设置$link_identifier，则使用上一个已经打开的数据库连接。调用后 mysql_query()函数会向$link_identifier 关联的 MySQL 中执行$query 语句。执行成功时返回 true，出错时返回 false。

【实践案例 10-9】

使用 mysql_query()函数向 MySQL 发送各种 SQL 语句，像创建库、打开库、创建表和插入数据等，实现代码如下所示：

```
<h2>使用 mysql_query()函数</h2>
<ul>
<?php
```

```
$conn=mysql_connect("localhost","root","123456");
if($conn)
{
    echo "<li>(1)成功建立连接。</li>";
    $str="create database temp;";           //创建 temp 数据库
    if(mysql_query($str,$conn))
    {
        echo "<li>(2)成功创建 temp 数据库。</li>";
        $str="use temp;";                   //打开 temp 数据库
        if(mysql_query($str))
        {
            echo "<li>(3)成功选择 temp 数据库。</li>";
            $str=<<<SQL                     //创建 users 数据表
            CREATE TABLE users (
              id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
              uname VARCHAR(45) NOT NULL,
              upass VARCHAR(45) NOT NULL,
              PRIMARY KEY ('id')
            )
SQL;
            if(mysql_query($str))
            {
                echo "<li>(4)成功创建 users 数据表。</li>";
                                            //向 users 表插入数据
                $str="insert into users(uname,upass) values
                ('admin','admin'),('itzcn','itzcn');";
                if(mysql_query($str))       //执行 SQL 语句
                {
                    echo "<li>(5)成功向 users 数据表插入两条数据。</li>";
                }
                mysql_query("select * from users;");//查询 users 表的数据
                echo "<li>(6)执行查询共影响".mysql_affected_rows()."行。
                </li>";
            }
        }
    }
    if(mysql_close($conn)) echo "<li>(7)数据库已经关闭。</li>";   //关闭连接
}
?></ul>
```

如上述代码所示，这里使用 mysql_query()函数执行了 5 条 SQL 语句，运行效果如图 10-30 所示。

```
错误描述：Access denied for user 'root'@'localhost' (using password: NO)
```

10.6.2 获取数据库信息

如果要判断一个数据库是否已经存在，需要先获取所有数据库的列表然后再比较。mysql 库提供了 mysql_list_dbs()函数和 mysql_db_name()函数来获取数据库信息。

1. mysql_list_dbs()函数

mysql_list_dbs()函数可以返回一个包含了当前 MySQL 中所有可用数据库的结果集，语法格式如下所示：

```
resource mysql_list_dbs ( [resource $link_identifier])
```

【实践案例 10-11】

使用 mysql_list_dbs()函数从 MySQL 服务器中获取所有数据库信息，并显示数据库名称。实现代码如下所示：

```
<h2>获取所有数据库信息</h2>
<ul><?php
$conn=mysql_connect("localhost","root","123456");
if($conn){
    $result=mysql_list_dbs($conn);              //获取数据库列表
    echo "<ul>";
    while ($db=mysql_fetch_assoc($result)) {
        echo "<pre>";
        print_r($db);                           //输出每一行结果
        echo "</pre>";
    }
    echo "</ul>";
    mysql_close($conn);
}?> </ul>
```

将上述代码保存为 mysql_list_dbs.php，在浏览器中执行效果如图 10-32 所示。

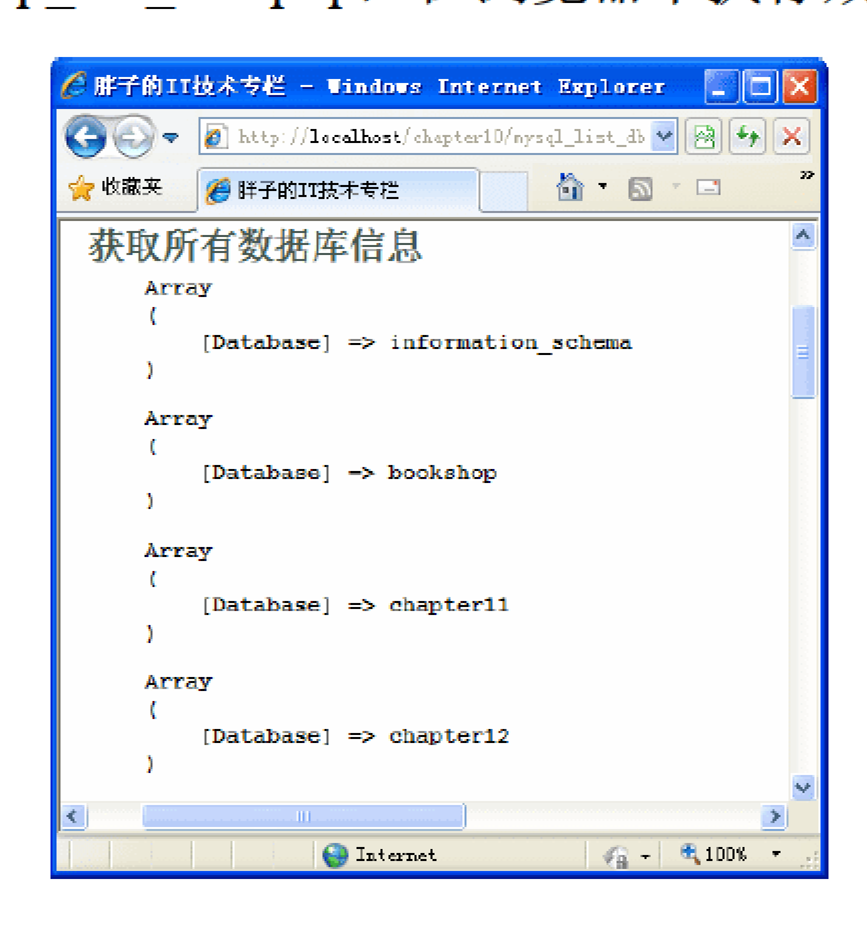


图 10-32　查看所有数据库

10.6.3 获取数据表信息

使用 mysql_list_tables()函数和 mysql_table_name()函数可以获取 MySQL 中数据表的信息，下面详细介绍它们：

1. mysql_list_tables()

mysql_list_tables()接受一个数据库名并返回它包含的所有表，语法格式如下所示：

```
resource mysql_list_tables ( string database [, resource link_identifier] )
```

其中，$database 参数表示要获取数据表列表的数据库名，$link_identifier 参数表示数据库连接标识符。它返回一个指定数据库中所有数据表的结果集，可使用类似 mysql_fetch_row()函数获取该结果集，若失败则返回 false。

下面的代码演示了如何使用 mysql_list_tables()函数获取 db_blog 数据库中的所有表名：

```
<?php
    $dbname = "db_blog";                        //指定要获取表名的数据库名称
    mysql_connect("localhost","root","123456");     //建立连接
    $result = mysql_list_tables($dbname);           //获取所有表
    while ($row = mysql_fetch_row($result)) {       //遍历输出表名称
        echo "表名: $row[0]\n";
    }
    mysql_free_result($result);                     //释放资源
?>
```

在这里要注意，使用 mysql_list_tables()函数后必须调用 mysql_free_result()函数释放结果集。

2. mysql_table_name()

mysql_table_name()函数接受 mysql_list_tables()返回的结果集，以及一个整数索引并返回表名。语法格式如下：

```
string mysql_tablename ( resource $result , int $i )
```

下面的示例代码使用 mysql_table_name()函数同样实现了获取 db_blog 数据库中的所有表名。

```
<?php
    $dbname = "db_blog";                        //指定要获取表名的数据库名称
    mysql_connect("localhost","root","123456");     //建立连接
    $result = mysql_list_tables($dbname);           //获取所有表
    for($index=0;$index<mysql_num_rows($result);$index++)//遍历输出表名称
    {
        echo "表名: ".mysql_table_name($result,$index)."\n";
```

```
    }
    mysql_free_result($result);                    //释放资源
?>
```

如上述代码所示，使用 mysql_num_rows()函数来获取结果集中表的总数。然后在使用 for 语句遍历时使用 mysql_table_name()函数获取表的名称。

10.6.4 获取列信息

除了获取数据库和表信息之外，mysql 库还提供了用于获取数据表中列信息的函数，像列的名称、数据类型以及是否主键等。

1. mysql_fetch_field()函数

mysql_fetch_field()函数可以用来从某个查询结果中取得列的信息，语法格式如下所示：

```
object mysql_fetch_field ( resource $result [, int $field_offset] )
```

$field_offset 参数表示列偏移量，如果没有指定则自动提取下一个未被提取的列。mysql_fetch_field()函数将返回一个对象描述列的各种信息，该对象包括表 10-1 列出的属性。

表 10-1　列属性表

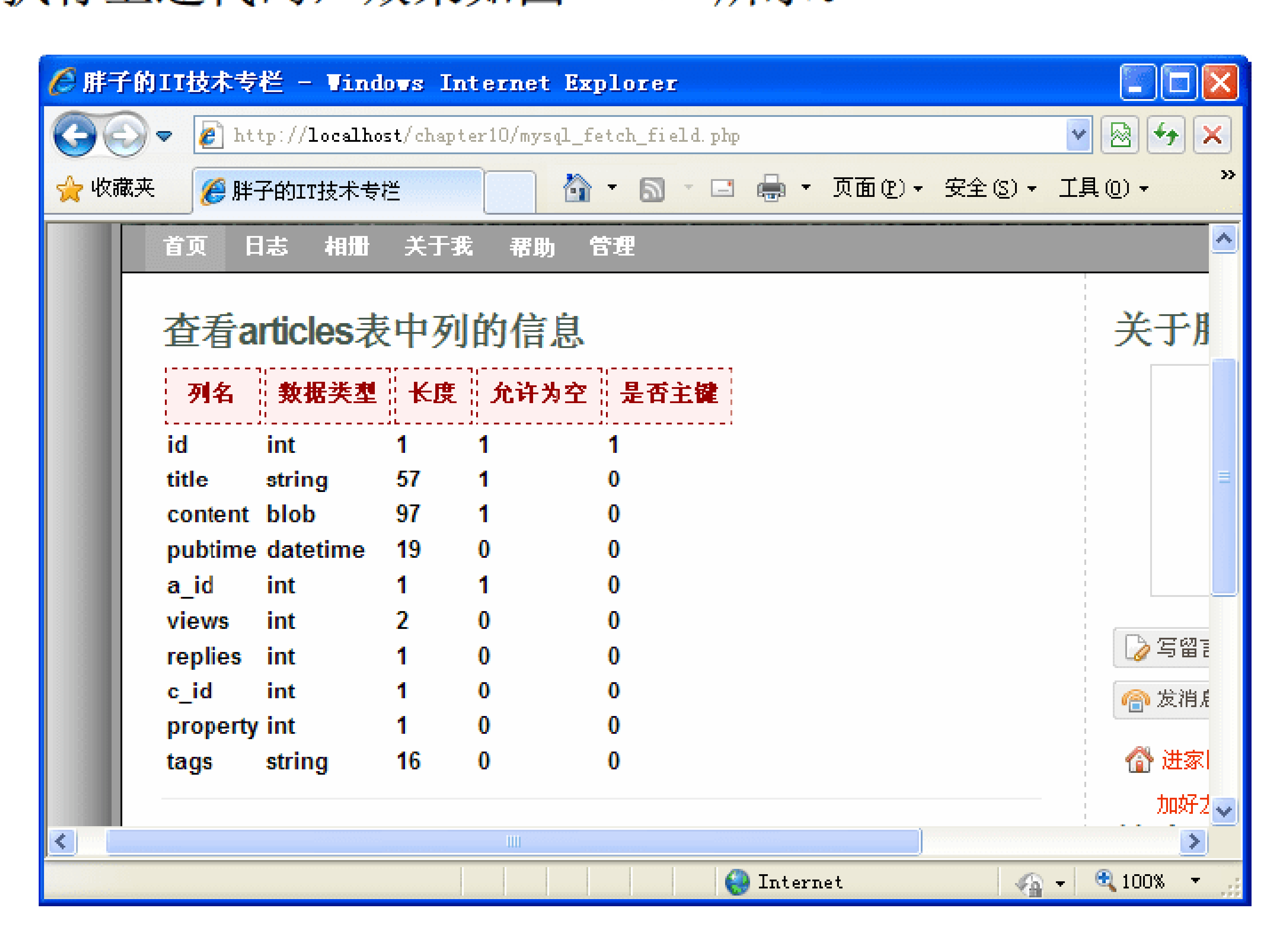
名称	说明
name	列名
table	所在的表名
max_length	最大长度
not_null	如果该列不能为 NULL，则为 1；否则为 0
primary_key	如果该列是 primary key 则为 1；否则为 0
unique_key	如果该列是 unique key 则为 1；否则为 0
multiple_key	如果该列是 non-unique key 则为 1；否则为 0
numeric	如果该列是 numeric 则为 1；否则为 0
blob	如果该列是 BLOB 则为 1；否则为 0
type	该列的数据类型
unsigned	如果该列是无符号数则为 1；否则为 0
zerofill	如果该列是 zero-filled（零填充）则为 1；否则 0

下面的代码使用 mysql_fetch_field()函数获取 articles 表中的所有列信息，包括列名、数据类型、长度、是否允许为空和是否主键。

```
<h2>查看 articles 表中列的信息</h2>
<table>
<tr>
<th>列名</th><th>数据类型</th><th>长度</th><th>允许为空</th><th>是否主键</th>
</tr>
<?php
```

```
    $conn = mysql connect('localhost', "root", "123456");
    mysql select db("db blog");
    $result=mysql_query("select * from articles");
    while($field=mysql_fetch_field($result))        //获取列的信息
    {
        echo "<tr>";
        echo "<td>".$field->name."</td>";           //调用 name 属性
        echo "<td>".$field->type."</td>";
        echo "<td>".$field->max length."</td>";
        echo "<td>".$field->not null."</td>";
        echo "<td>".$field->primary key."</td>";
        echo "</tr>";
    }
    mysql close();                                  //关闭连接
?></table>
```

如上述代码所示，这里仅使用了 name、type、max_length、not_null 和 primary_key 属性。在浏览器中执行上述代码，效果如图 10-34 所示。



列名	数据类型	长度	允许为空	是否主键
id	int	1	1	1
title	string	57	1	0
content	blob	97	1	0
pubtime	datetime	19	0	0
a_id	int	1	1	0
views	int	2	0	0
replies	int	1	0	0
c_id	int	1	0	0
property	int	1	0	0
tags	string	16	0	0

图 10-34　获取表中列的信息

2. mysql_num_fields()函数

mysql_num_fields()函数可以返回结果集中列的数量，语法格式如下所示：

```
int mysql_num_fields ( resource $result )
```

下面的代码演示了如何使用 mysql_num_fields()函数获取 articles 表中列的数量：

```
<?php
$con = mysql connect("localhost", "root", "123456");
mysql select db("db blog");
$result = mysql_query("select * from articles");
```



图 10-35　获取 articles 表中的列名

下面演示了使用 mysql_field_flags()函数获取 articles 表中列属性的实现代码：

```
<h2>使用 mysql_field_flags()获取 articles 表中的列</h2>
<?php
    $con = mysql_connect('localhost', 'root', '123456');
    //获取 db_blog 数据库中 articles 表的列信息
    $fields = mysql_list_fields("db_blog", "articles", $con);
    $columns = mysql_num_fields($fields);                //获取列的数量
    echo "<p>表 articles 中共有".$columns."个列，分别是：</p>";
    echo "<ul>";
    for ($i = 1; $i <=$columns; $i++)
    {
        echo "<li>第".$i."个列属性是：".mysql_field_flags($fields, $i-1) .
        "</li>";                                  //获取列名
    }
    echo "</ul>";
?>
```

在浏览器中执行上述代码，效果如图 10-36 所示。

5．mysql_field_len()函数

mysql_field_len()函数可以返回指定列的长度，如果失败则返回 false。该函数语法格式如下所示：

```
int mysql_field_len ( resource $result, int $field_offset )
```

mysql_field_len()函数的使用方法比较简单。

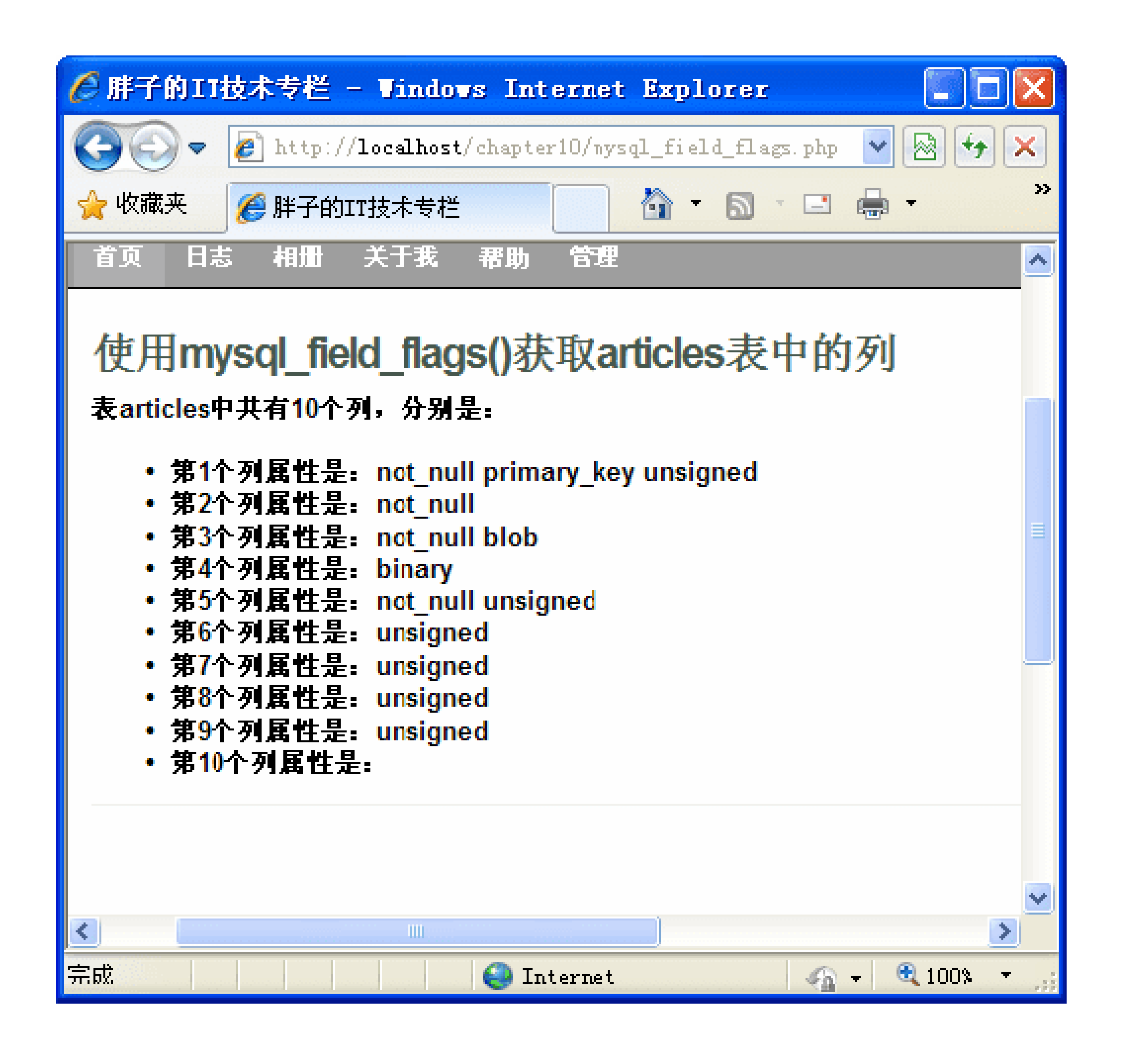


图 10-36　获取 articles 表中列的属性

6. mysql_field_name()函数

mysql_field_name()可以返回指定索引的列名，语法格式如下所示：

```
string mysql_field_name ( resource $result, int $field_index )
```

其中，$result 参数必须是一个合法的结果标识符；$field_index 参数是该字段从 0 开始的数字偏移量。

下面的示例代码演示了使用 mysql_field_name()函数获取 articles 表中第 2 列和第 4 列的名称：

```
<?php
    $con = mysql_connect('localhost', 'root', '123456');
    mysql_select_db("db_blog", $con);
    $result=mysql_query("select * from articles");
    echo"articles 表中第 2 列的名称是:".mysql_field_name($result,1)."<br/>";
    echo "articles 表中第 4 列的名称是: ".mysql_field_name($result,3);
?>
```

上述代码执行后的输出结果如下：

```
articles 表中第 2 列的名称是: title
articles 表中第 4 列的名称是: pubtime
```

7. mysql_field_type()函数

mysql_field_type()函数的功能和 mysql_field_name()函数相似，参数也完全相同。但 mysql_field_type()函数返回的是列类型而不是列名，列可以是 int、real、string、blob 类型以及其他类型。

8. mysql_field_table()函数

mysql_field_table()函数返回指定结果集中指定偏移量列所在的表名，语法格式如下所示：

```
string mysql_field_table ( resource $result, int $field_offset )
```

10.7 使用 mysqli

在前面介绍的都是有关 mysql 库的操作，PHP 还提供更高级的 mysqli 库。mysqli 库不仅实现了 mysql 库的所有操作，而且支持以面向对象的方式操作数据库，同时有很多新增功能。

本节将详细介绍如何使用 mysqli 库操作 MySQL，像建立连接、执行查询语句、获取结果集以及预处理语句等。

10.7.1 基本操作

mysqli 库被启用之后就可以直接使用该库里面的函数了。mysqli 库里面的函数通常都以“mysqli_”为前缀，表 10-2 列出了其中常用函数及其功能。

表 10-2 扩展包常用函数

函数名称	功能	相似 mysql 函数
mysqli_connect()	连接 MySQL 服务器	mysql_connect()
mysqli_select_db()	选择一个数据库	mysql_select_db()
mysqli_close()	关闭数据库连接	mysql_close()
mysqli_query()	执行 SQL 查询语句，返回结果集	mysql_query()
mysqli_num_rows()	获取结果集中受影响的行数	mysql_num_rows()
mysqli_error()	产生错误消息，如果没有错误产生一个空的字符串	mysql_error()
mysqli_errno()	产生错误的所在位置，即行号	mysql_errno()

【实践案例 10-13】

使用表 10-2 中列出的 mysqli 库函数建立一个到 MySQL 的连接，然后选择 db_blog 数据库并将 articles 表的行数输出。实现代码如下所示：

```
<h2>使用 mysqli 库连接 MySQL</h2>
<?php
$conn=mysqli_connect("localhost","root","123456");       //建立连接
if($conn)
{
    echo "(1)建立到 MySQL 数据库的连接。<br/>";
    mysqli_select_db($conn,"db_blog");                    //选择数据库
```

```
        echo "(2)已经连接到 db_blog 数据库。<br/>";
        $result=mysqli_query($conn,$strsql);                  //执行查询
        echo "(3)执行 SQL 语句查询 articles 表所有行。<br/>";
        $rows=mysqli_num_rows($result);                       //获取查询的结果数
        echo "(4)articles 表中当前共有: $rows 条<br/>";
        mysqli_close($conn);                                  //关闭连接
        echo "(5)关闭数据库连接。";
    }else{
        echo "建立连接失败! <br/>";
        echo "错误编码: ".mysqli_errno()."<br/>";              //输出错误编码
        echo "错误描述: ".mysqli error();                      //输出错误信息
    }
    ?>
```

如上述代码所示，大部分的 mysqli 库函数与 mysql 库中的函数使用方法相同，但是这里要注意 mysqli_select_db()函数和 mysqli_query()函数两个参数的顺序与 mysql 库相反。代码执行效果如图 10-37 所示。



从上面的案例可以看出，使用 mysqli 库连接数据库的方式与前面的 mysql 库中没有什么大的区别，只是个别函数的调用有些差异。

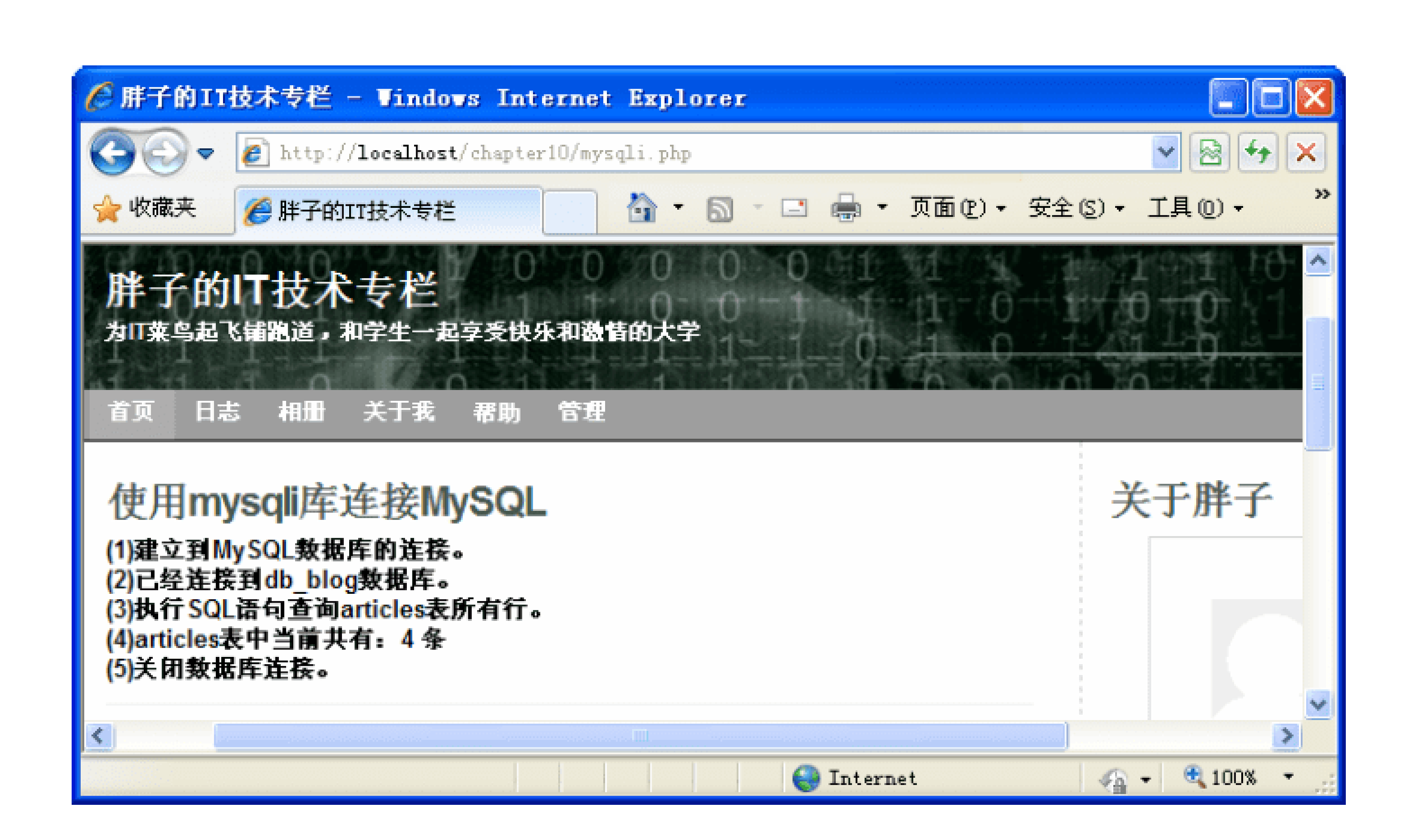


图 10-37　使用 mysqli 库连接 MySQL

10.7.2　获取结果集

使用 mysqli_num_rows()可以获取 mysqli_query()执行结果集中的行数，如果要遍历结果集就需要使用获取函数。与 mysql 库一样，mysqli 库同样可以获取一行、数组或者一个对象，这些常用函数如表 10-3 所示。

mysqli_fetch_object()函数遍历 articles 表并输出 id、title 和 views 列，运行效果如图 10-39 所示。

```
<h2>显示所有文章信息</h2>
<table>
<tr><th>编号</th><th>标题</th><th>浏览次数</th></tr>
<?php
$conn=mysqli_connect("localhost","root","123456");
if($conn)
{
    mysqli_select_db($conn,"db_blog");
    $strsql="select id,title,views from articles order by id";//查询语句
    $result=mysqli_query($conn,$strsql);                  //执行查询
    while($row= mysqli_fetch_object($result)){            //获取一行的对象
        echo "<tr>";
        echo "<td>".$row->id."</td>";
        echo "<td>".$row->title."</td>";
        echo "<td>".$row->views."</td>";
        echo "</tr>";
    }
    mysqli_close($conn);                                  //关闭连接
}
?></table>
```



图 10-39　使用 mysqli_fetch_object()函数

10.7.3　使用预处理语句

通常在一些 Web 应用中要操作一些表，只是每次的参数不同。如果访问量很多时，这些语句对系统的使用会很大。因此， MySQL 4.1 以后版本提供了对预处理语句的使用，实现较低的占用及较少的代码来实现任务。

为了使用 MySQL 的预处理语句功能，mysqli 库提供了 prepare 系列的函数保障开发人员在执行语句时的稳定性和安全性，以及代码编写的简洁性。

按功能来分，mysqli 库可以分为绑定参数预处理语句和绑定结果预处理语句。

1. 绑定参数预处理语句

绑定参数预处理语句允许创建一个查询模板并保存在 MySQL 上。当开始一个查询时，需要把一些数据参数发给 MySQL 以填充这个模板。当 MySQL 认为这个查询是完整无误的查询后则立即执行。

例如，对于一个类似如下的条件查询语句：

```
select * from articles where id=2;
```

使用查询模板后可以修改为如下形式：

```
select * from articles where id=?;
```

这里的“?”表示语句中的参数占位符，执行时将被实际值所代替。一个语句中可以同时包含多个参数占位符，例如如下的插入语句：

```
insert into articles(id,title,views,tags) values(?,?,?,?);
```

表 10-4 中列出了使用绑定参数预处理语句需要用到的函数。

表 10-4　绑定参数预处理语句函数

方法名称	过程化语法格式	参数描述	功能说明
mysqli_stmt_prepare()	mysqli_stmt_prepare(mysqli_stmt stmt , string query)	stmt 表示一个预处理语句对象；query 表示一条 SQL 语句	准备要执行的语句
mysqli_stmt_execute()	mysqli_stmt_execute(mysqli_stmt stmt)	stmt 表示一个预处理语句对象	执行预处理语句，该语句何时执行，取决于语句类型
mysqli_stmt_close()	mysqli_stmt_close(mysqli_stmt stmt)	同上	关闭预处理语句占用的资源
mysqli_stmt_bind_param()	mysqli_stmt_bind_param(mysqli_stmt stmt, string types , mixed &var1 [, mixed &...])	stmt 表示一个预处理语句对象；types 参数表示其后各个变量($var1 ...$varn)的数据类型，types 绑定的变量的数据类型接受 4 个参数：I 表示 integer 类型；D 表示 double 类型；S 表示 string 类型；B 表示 blob 类型	将变量名绑定到相应的字段（绑定参数）
mysqli_stmt_affected_rows()	mysqli_stmt_affected_rows (mysqli_stmt stmt)	stmt 表示一个预处理语句对象	返回预处理语句执行后影响的行数

下面使用上表列出的函数创建一个示例，实现使用预处理语句向 articles 表中插入一行

数据。具体代码如下所示：

```
<?php
$conn=mysqli_connect("localhost","root","123456");
if($conn)
{
    mysqli_select_db($conn,"db_blog");
    $strsql="insert into articles(id,title,views) values(?,?,?);";
                                                    //定义预处理语句
    $stmt = mysqli_prepare($conn, $strsql);     //创建预处理对象
    mysqli_stmt_bind_param($stmt,"isi",$id,$title,$views);
                                                    //为预处理语句绑定参数
    $id=6;                                          //指定$id 参数的值
    $title="test";
    $views=3;
    mysqli_stmt_execute($stmt);                     //执行预处理语句
    $rows=mysqli_stmt_affected_rows($stmt);     //获取影响的行数
    echo "本次操作一共影响 : $rows 行 <br/>";       //输出
    mysqli_close($conn);                            //关闭连接
}
?>
```

这里要注意，在 mysqli_prepare()函数创建的预处理对象中的参数数量必须与 mysqli_stmt_bind_param()函数绑定的参数数量相同。本例中的预处理语句有 3 个参数，分别表示 id、title 和 views。

调用 mysqli_stmt_bind_param()时第 1 个参数是预处理对象，第 2 个参数是一个字符串用于指定占位符的数据类型，后面是每个参数对应的变量。第 2 个参数是如下类型标识的组合。

- **i　表示占位符对应的类型是 int 类型。**
- **b　表示占位符对应的类型是 blobs 类型。**
- **d　表示占位符对应的类型是 double 或者 float 类型。**
- **s　表示占位符对应的类型是字符串或者其他类型。**

这里使用的是 isi 组合，表示要插入的数据类型依次为 int、字符串和 int。此时每个参数的类型都已准备好，然后为参数指定值。再使用 mysqli_stmt_execute()函数发送给服务器处理，最后关闭连接。

2．绑定结果预处理语句

使用绑定结果的预处理语句允许将 PHP 脚本中的变量绑定到所获取的相应字段上，从而可以在有时难以处理的索引或者关联数组的结果集中提取值，然后在必要时使用这些变量。

使用绑定结果预处理语句时，除上面的函数之外还需要使用 mysqli_stmt_bind_result()函数和 mysqli_stmt_fetch()函数。

mysqli_stmt_bind_result()方法将变量绑定到所获取的字段上，该函数的语法格式如下：

```
bool mysqli_stmt_bind_result ( mysqli_stmt stmt, mixed &var1 [, mixed &...] )
```

mysqli_stmt_fetch()方法将所获取的字段绑定到一个变量上，该函数的语法格式如下：

```
bool mysqli_stmt_fetch ( mysqli_stmt stmt )
```

下面创建一个示例，使用绑定结果准备语句查询数据，代码如下：

```
<?php
$conn=mysqli_connect("localhost","root","123456");
if($conn)
{
    mysqli_select_db($conn,"db_blog");
    $strsql="select id,title,views from articles";     //查询语句
    $stmt = mysqli_prepare($conn, $strsql);              //创建预处理对象
    mysqli_stmt_execute($stmt);                          //执行查询语句
    mysqli_stmt_bind_result($stmt,$id,$title,$views);    //对结果集绑定变量
    while (mysqli_stmt_fetch($stmt)) {                   //遍历输出每个变量
        echo $id." ";
        echo $title." ";
        echo $views." <br/>";
    }
    mysqli_close($conn);                                 //关闭连接
}
?>
```

在这里要注意 mysqli_stmt_bind_result()函数第 1 个参数是预处理对象，后面的每个变量按顺序依次对应结果集中的一列。mysqli_stmt_fetch()函数会对结果集中的每行数据进行读取和绑定变量，因此循环结束后会遍历整个结果集。

10.8　项目案例：实现基于数据库的留言本

在第 9 章的项目案例中实现了一个留言本，不过是使用 Cookie 来存储的。有很多局限性，例如如果客户端禁用 Cookie 将无法保存，Cookie 过期之后数据会自动丢失，等等。

为了使留言数据可以持久有效，本案例将使用 MySQL 数据库进行存储，并重写留言本，而且在本案例中将为留言本添加管理员回复的功能。

【实例分析】

mysqli 库是对 mysql 库的扩展和增强，在前面的介绍中作为与 mysql 库的对比，仅讲解了 mysqli 库的过程式使用方法。mysqli 库还支持以面向对象的方式操作数据库，我们将在本案例中使用这种方式，具体步骤如下所示。

（1）首先使用命令以 root 用户连接到 MySQL。

（2）执行如下命令创建留言本实例数据库：

```
}
//每页数量
$page_size = PAGESIZE;
//获取总数据量
$sql = "select count(*) as amount from tb_messages";
$result = $conn->query($sql);
$row = $result->fetch_row();
$amount = $row[0];
//记算总共有多少页
if( $amount ){
    if( $amount < $page_size ){ $page_count = 1; }
                                    //如果总数据量小于$PageSize，那么只有一页
    if( $amount % $page_size ){             //取总数据量除以每页数的余数
        $page_count = (int)($amount / $page_size) + 1;
                        //如果有余数，则页数等于总数据量除以每页数的结果取整再加一
    }else{
        $page_count = $amount / $page_size;
                        //如果没有余数，则页数等于总数据量除以每页数的结果
    }
}
else{
    $page_count = 0;
}
$result->close();
//翻页链接
$page_string = '';
if( $page == 1 ){
    $page_string .= '第一页|上一页|';
}
else{
    $page_string .= '<a href=?page=1>第一页</a>|<a href=?page='.($page-1).'>
    上一页</a>|';
}
if( ($page == $page_count) || ($page_count == 0) ){
    $page_string .= '下一页|尾页';
}
else{
    $page_string .= '<a href=?page='.($page+1).'>下一页</a>|<a href=
    ?page='.$page_count.'>尾页</a>';
}
//获取数据，以二维数组格式返回结果
if( $amount ){
    $sql = "select * from tb_messages order by id desc limit ".
    ($page-1)*$page_size .", $page_size";
    $result = $conn->query($sql);
```

```
    while ( $row = $result->fetch_assoc() ){
        $rowset[] = $row;
    }
}else{
    $rowset = array();
}
$result->close();
?>
```

（11）有了上面的代码，剩下的工作就是遍历结果集并输出显示。这部分代码非常简单，这里就不再介绍，请参考源代码。

（12）参考如图 10-42 所示的效果创建管理员登录页面 login.php。

（13）创建 loginCheck.php 文件对管理员信息进行验证。如果成功，则在 Session 中保存管理员，并转向首页 index.php。

（14）在首页的留言列表中单击“回复”链接进入 reply.php 页面回复留言。该页面只有管理员能进行操作，所以需要对权限进行判断。实现代码如下：

```
if(!checkSession())
{
    showmsg("login.php","请先登录后再操作!");
}
```

（15）在 reply.php 页面只需要输入回复内容，单击“保存”按钮即可。保存成功后将看到对话框，如图 10-43 所示。

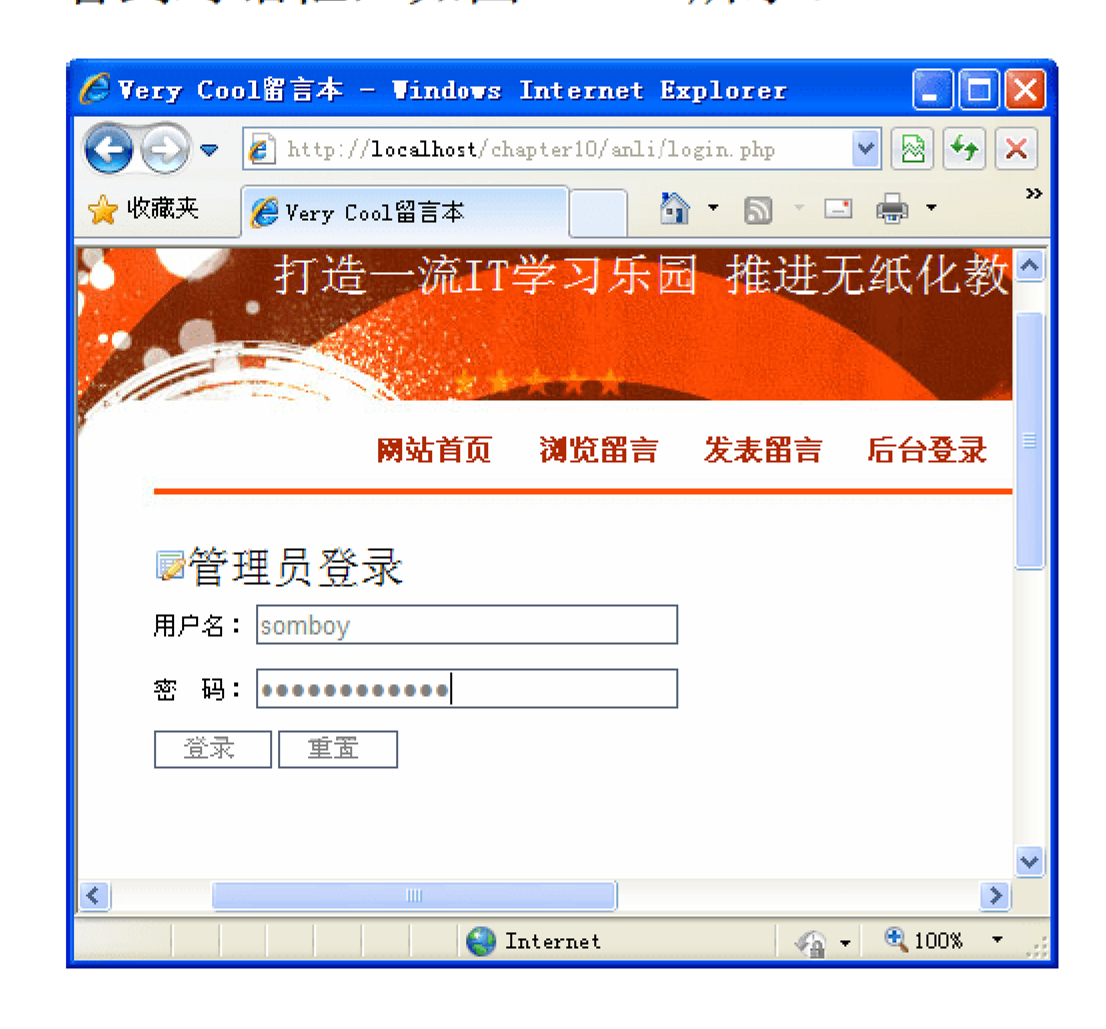




图 10-42 登录页面

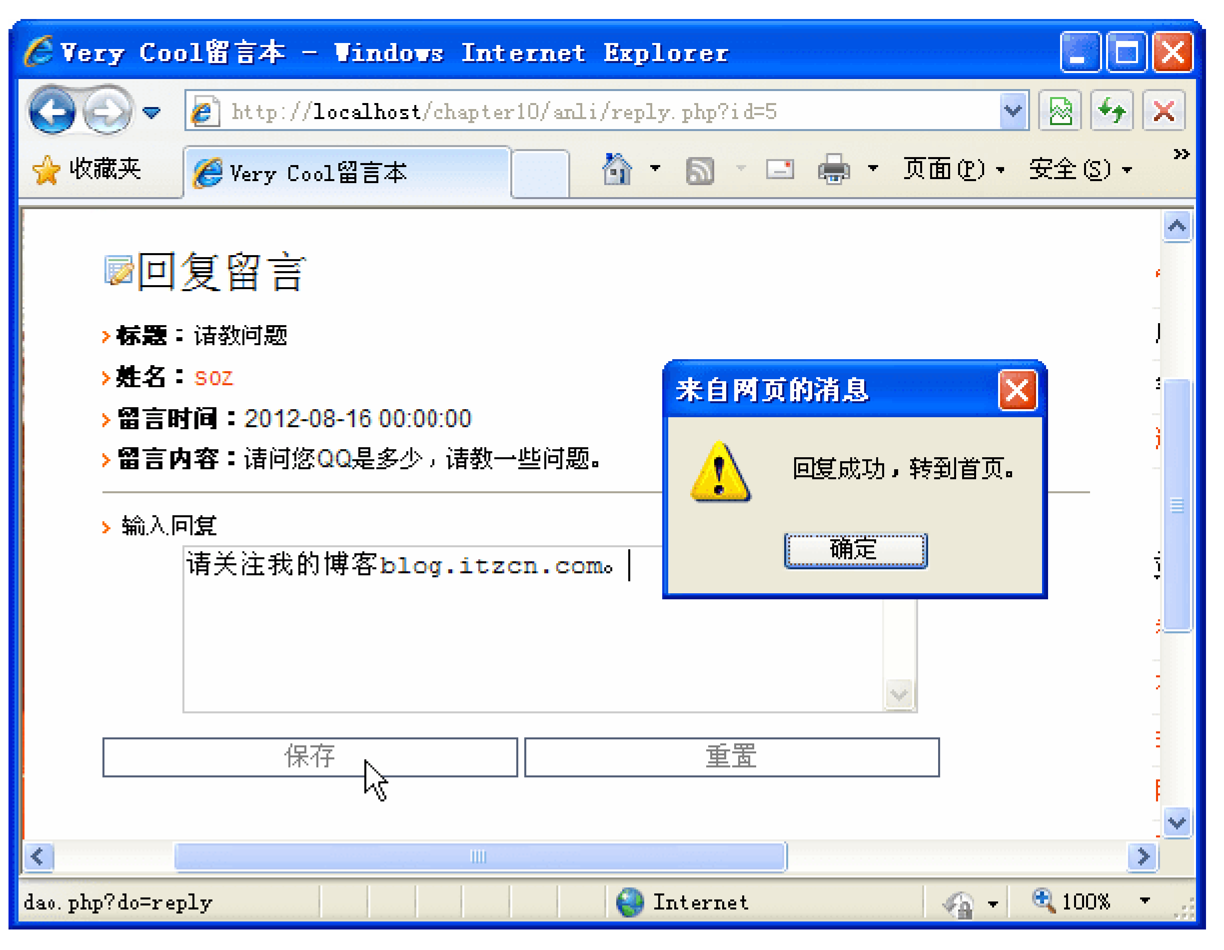


图 10-43 回复页面

（16）在 reply.php 页面输入的回复留言会提交到 dao.php 文件。如下所示是 dao.php 文件中的接受和保存代码：

```
//回复操作
if($do=='reply') {
    $reply=$_POST["reply"];                    //回复内容
```

```
    $query="update tb_messages set reply='".$reply."' where id=".$_POST["msgid"];
    $conn->query($query);                    //执行 SQL 语句
    $conn->close();                          //关闭连接
    showmsg("index.php","回复成功，转到首页。");
}
```

（17）回复成功后会再次跳转到首页，此时将看到刚才输入的回复信息，如图 10-44 所示。

在如图 10-44 所示的留言首页右侧还可以看到当前管理员的用户名、密码和“退出”链接，这些内容在登录之前是不显示的。至此，有关留言本的核心代码就介绍完了。



图 10-44　回复后首页

10.9　习题

一、填空题

（1）在使用 mysql_connect()建立连接时指定__________常量可以使用 SSL 加密连接。

（2）如果要创建持久性连接需要使用__________函数。

（3）__________函数可以列出 MySQL 服务器中所有的数据库。

（4）假设数据表 user 中有 3 个列，分别是 uid、uname 和 upass。在下面的空白处填写合适的代码，使程序运行正确。

```
<?php
$strsql="select * from tab_user";
$result=mysql_query($strsql);
while($row=           )
{
```

```
    echo "编号: ".$row['uid'].",用户名: ".$row['uname'].", 密码: ".$row
    ['upass']."<br/>";
}
?>
```


（5）对于 mysql_fetch_field()函数的返回调用__________属性可以获取列的数据类型。

（6）在下面代码的空白处填写代码，使程序可以显示结果集中的行数。

```
<?php
$conn=mysqli connect("localhost","root","123456");
mysqli select db($conn,"db blog");
$result=mysqli_query($conn, "select * from table");
$rows= ___________ ($result);
echo "表中当前共有: $rows 条<br/>";
?>
```

二、选择题

（1）下面关于 MySQL 语句错误的是__________。

A．select now();select user();

B．select now(),select user();

C．show databases;

D．quit

（2）下面关于 mysqli 库的描述错误的是__________。

A．支持本地绑定、占位符和游标

B．支持面向对象的调用方式

C．由 PHP 自动集成

D．支持预处理语句

（3）如果要从 MySQL 中获取字符集的名称，应该使用__________函数。

A．mysql_client_encoding()

B．mysql_get_host_info()

C．mysql_stat()

D．mysql_get_client_info()

（4）下面关于预处理的描述错误的是__________。

A．使用 mysqli_prepare()函数创建预处理对象

B．使用 mysqli_stmt_bind_param()函数绑定参数

C．使用 mysqli_stmt_execute()函数执行预处理语句

D．使用 mysqli_num_rows()函数获取结果集中的行数

（5）假设 user 表中有 5 个字段，执行下面的代码，输出的结果是__________。

```
$sql = "SELECT username , password , sex , age FROM user";
$result = mysql_query($sql,$con);   //执行查询操作
echo mysql_num_fields($result);     //输出表中字段数量
```

A．2
B．3
C．4
D．5

三、上机实践

1．使用 mysql 库操作学生信息

本次上机练习要求读者使用本章介绍的 mysql 库完成，具体要求如下。

（1）在 MySQL 中创建一个 students 数据库。

（2）向 students 数据库中创建一个学生信息表 stuinfo，包含的列有 stuid、name、sex、class、phone、address、postcode。

（3）向 stuinfo 表中执行如下 insert 语句插入一条学生信息。

```
INSERT INTO stuinfo(stuid,name,sex,class,phone,address,postcode)
VALUES(1001,'zht',1,'computer','12345678901','beijing','100086');
```

（4）查询是否有来自 beijing 的学生信息。

（5）获取数据库的字符编码集。

（6）最后捕捉错误，并关闭连接。

2．显示数据表字段的详细信息

本次上机要求读者使用本章所介绍的技术，实现在页面输出某个表中字段的详细信息的功能，效果类似于如图 10-45 所示的效果。



图 10-45 显示学生表字段信息

10.10 实践疑难解答

10.10.1 PHP+MySQL 文字乱码显示问题



php +mysql 文字乱码显示问题
网络课堂：http://bbs.itzcn.com/thread-19698-1-1.html

【问题描述】：PHP 和 MySQL 数据库编程的初学者遇到最多的问题可能就是乱码。这里就再总结一下 PHP 中常见的乱码解决方案，希望对大家有帮助。

【正确回答】：PHP 最终生成的是 HTML 文本文件，但它要读取数据库里的文本，或将文本保存到数据库。由于 MySQL 支持多字符集，但默认情况下，MySQL 不知道 PHP 提交的是什么编码的文本，因此不兼容时发生转换出错产生乱码。

以 UTF-8 编码为例，PHP 要输出头："header("Content-Type: text/html; charset= UTF-8")"，静态页面添加"<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">"。所有文件的编码格式为 UTF-8，可用记事本打开，另存为选择编码为 UTF-8，覆盖源文件。

PHP 与数据库的编码必须一致。可以修改 MySQL 配置文件 my.ini，以 utf8 编码为例，配置如下：

```
[mysql]
default-character-set=utf8
[mysqld]
default-character-set=utf8
default-storage-engine=MyISAM
```

在上面 mysqld 小节中加入下面代码：

```
default-collation=utf8_bin
init_connect='SET NAMES utf8'
```

然后，在数据库操作前使用"mysql_query("set names ' utf8'");"语句使两者编码一致，注意这里没有短横线。这样插入和查询数据时就不会出现乱码了。

10.10.2 缺少 mysqli 扩展的问题



缺少 mysqli 扩展的问题
网络课堂：http://bbs.itzcn.com/thread-19699-1-1.html

【问题描述】：下载了一个 MySQL 管理工具 phpmyadmin，但是在本机部署却提示"缺少 mysqli 扩展。请检查 PHP 配置。"错误，具体如图 10-46 所示。

127.0.0.1/phpmyadmin/
phpMyAdmin - 错误
缺少 mysqli 扩展。请检查 PHP 配置。 <a href="Documentation.html#faqmysql" target="documentation">

图 10-46　错误提示

在网上找了很多解决办法，试了也不管用。

【解决办法】：主要原因是 phpmyadmin 依赖 mysqli 库，而当前运行的环境中没有所造成的。可以从如下几个方面检查：

（1）检查 php.ini 文件中“;extension=php_mysqli.dll”是不是已经启用，也就是去掉前面的分号。

（2）检查 php.ini 文件中 extension_dir 值的地址是不是指向了 PHP 下的 ext 目录，这一步是关键，而且 php_mysqli.dll 文件应该位于这个地址中。也可以绝对路径，例如 extension_dir = "e:\www\php54\ext"。

（3）检查 lib_mysql.dll 有没有复制到 Windows 目录下。这个 dll 文件有多种形式，有加下划线的 libmysql_d.dll，因此注意要自己看。

（4）检查 PHP 安装目录 ext 目录下，php_mysqli.dll 文件是不是存在。如果没有，则需要下载或者复制一个。

（5）查看 PHP 的 phpinfo()函数，在“详细”页面看看 mysql 和 mysqli 是不是已经启动。这一步就是检查有没有开启 mysqli。没有开启的话，在“详细”页面中是找不到 mysqli 关键字的。

经过上面的几个步骤应该可以解决找不到 mysqli 库的问题。当然，也可以修改 phpmyadmin 让它使用 mysql 库。具体步骤就不说了，查看官方帮助就行。

第 11 章 PHP 高级开发

通过对本书前面内容的学习，读者已经掌握了使用 PHP 开发一般性网站的技术。但如果要构建一个用户体验好、高效且安全的网站，这些知识还远远不够。因为，这样的网站通常需要结合多种 PHP 技术，还会涉及一些高级应用。例如，为了程序安全，需要进行加密；为了维护方便，需要遵循开发规范；等等。

本章首先介绍如何在不影响用户操作的情况下使用 Ajax 进行异步刷新、发送 GET 和 POST 请求以及处理文本和 XML 的内容。然后讲解 PHP 的加密技术，像 md5 和 sha1 加密。在最后给读者列举了 PHP 开发时需要遵循的命名规范、代码规范和项目规范。

本章学习要点：

- **理解 Ajax 的通信原理**
- **掌握不同浏览器下 XMLHttpRequest 对象的创建方法**
- **熟悉 XMLHttpRequest 对象的属性和方法**
- **掌握 Ajax 中 GET 和 POST 发送数据的方法**
- **掌握处理文本和 XML 数据的方法**
- **掌握 PHP 内置加密函数的使用**
- **熟悉常用的加密扩展库**
- **熟悉常用的 PHP 开发编程规范**

11.1 使用 Ajax 异步通信

与传统的 Web 开发模式相比，Ajax 提供了一种以异步方式与服务器通信的机制。这种机制的最大特点就是不必刷新整个页面便可以对页面的局部进行更新。应用 Ajax 使客户端与服务器端的功能划分得更细，客户端只获取需要的数据，而服务器也只为有用的数据工作，从而大大节省了网络带宽、提高网页加载速度和运行效果。

11.1.1 Ajax 简介

Ajax 是 Asynchronous JavaScript And XML 的简称，中文含义为异步 JavaScript 和 XML。基于 Ajax 的开发与传统 Web 开发模式最大的区别就在于传输数据的方式不同，前者为异步，后者为同步。

Ajax 与传统的 Web 相比具有哪些优势呢。下面通过两张图来对比一下，其中图 11-1

所示为传统 Web 应用程序的工作原理。图 11-2 所示为 Ajax 程序的工作原理。

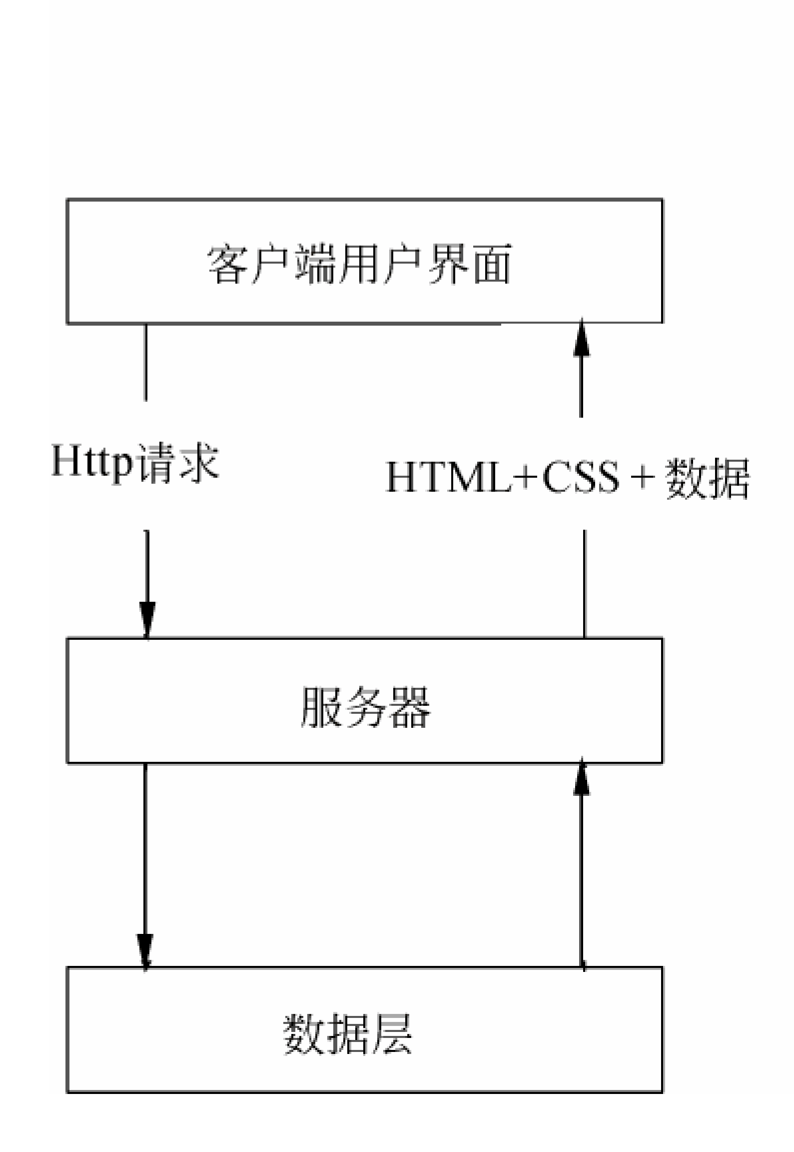


图 11-1　传统 Web 工作原理

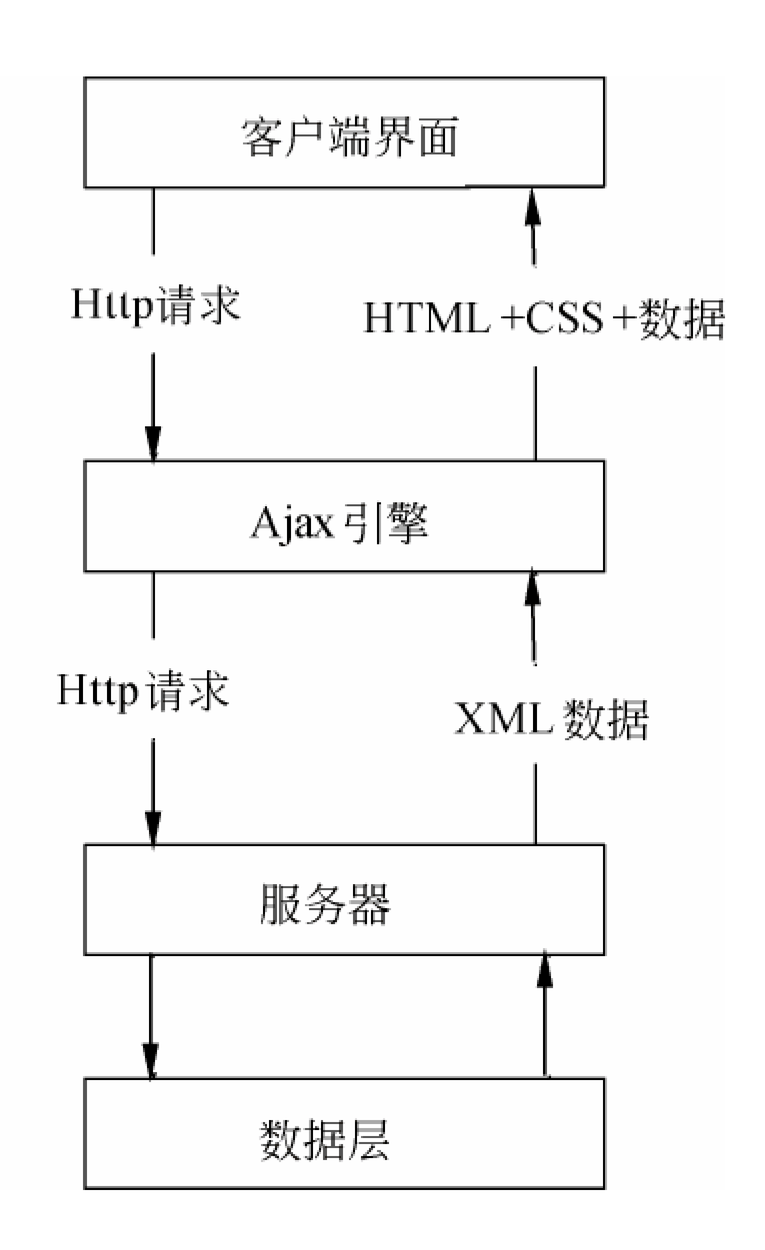


图 11-2　Ajax 工作原理

结果很明显：图 11-1 所示的传统方式在提交请求时，服务器承担大量的工作，客户端只有数据显示的功能。而图 11-2 所给出的 Ajax 方式中，客户端界面和 Ajax 引擎都是在客户端运行，这样大量的服务器工作可以在 Ajax 引擎处实现。

也就是说，与传统的 Web 应用不同，Ajax 采用异步交互过程。Ajax 在用户与服务器之间引入了一个中间介质，克服了网络交互过程中的处理和等待缺点。相当于在用户和服务器之间增加了一个中间层，使用户操作与服务器响应异步化。这把以前服务器负担的一些工作转移到客户端，利用客户端闲置的处理能力，减轻服务器和带宽的负担，从而达到节约服务器空间以及带宽租用成本的目的。

虽然 Ajax 如此先进，但它不是一项新技术，而是很多成熟技术的集合。主要包括：客户端脚本语言 JavaScript、异步数据获取技术 XMLHttpRequest、数据互换和操作技术 XML 和 XSLT、XHTML 和 CSS 显示技术等。

11.1.2　XMLHttpRequest 对象简介

XMLHttpRequest 对象是整个 Ajax 开发过程中的核心，它实现了与其他 Ajax 技术的结合。例如，发送请求、传递参数、获取响应以及处理结果等。

1. 创建 XMLHttpRequest 对象

XMLHttpRequest 对象并非最近才出现，最早在 Microsoft Internet Explorer 5.0 中将 XMLHttpRequest 对象以 ActiveX 控件的方式引入，被称为 XMLHTTP。其他浏览器（如 Firefox、Safari 和 Opera）将其实现为一个本地 JavaScript 对象。由于存在这些差别，在创建 XMLHttpRequest 对象实例时，JavaScript 代码中必须包含有关的逻辑，从而判断使用 ActiveX 技术或者使用本地 JavaScript 对象技术来创建 XMLHttpRequest 的一个实例。

readyState 属性值及其说明。

表 11-2　readyState 属性值

值	说明
0	表示未初始化状态，此时已经创建一个 XMLHttpRequest 对象，但是还没有初始化
1	表示发送状态，此时已经调用 open()方法，并且 XMLHttpRequest 已经准备好把一个请求发送到服务器
2	表示发送状态，此时已经通过 send()方法把一个请求发送到服务器端，但是还没有收到一个响应
3	表示正在接收状态，此时已经接收到 HTTP 响应头部信息，但是消息体部分还没有完全接收结束
4	表示已加载状态，此时响应已经被完全接收

通过属性可以了解 XMLHttpRequest 对象的状态，但如果要操作 XMLHttpRequest 对象，则需要通过它的方法。表 11-3 列出了该对象的常用方法及其说明。

表 11-3　XMLHttpRequest 对象常用方法

方法名称	说明
abort()	中止当前请求
open(method,url)	使用请求方式（GET 或 POST 等）和请求地址 URL 初始化一个 XMLHttpRequest 对象（这是该方法最常用的重载形式）
send(args)	发送数据，参数是提交的字符串信息
setRequestHeader(key,value)	设置请求的头部信息
getResponseHeader(key)	方法用于检索响应的头部值
getAllResponseHeaders()	方法用于返回响应头部信息（键/值对）的集合



getResponseHeader()和 getAllResponseHeaders()仅在 readyState 值大于或等于 3（接收到响应头部信息以后）时才可用。

3. XMLHttpRequest 对象工作流程

Ajax 程序主要通过 JavaScript 事件来触发，在运行时需要调用 XMLHttpRequest 对象发送请求和处理响应。客户端处理完响应之后，XMLHttpRequest 对象就会一直处于等待状态，这样一直周而复始的工作。

所以这个基本流程是：XMLHttpRequest 对象初始化→发送请求→服务器接收→服务器返回→客户端接收→修改客户端页面内容。只不过这个过程是异步的，如图 11-3 所示。

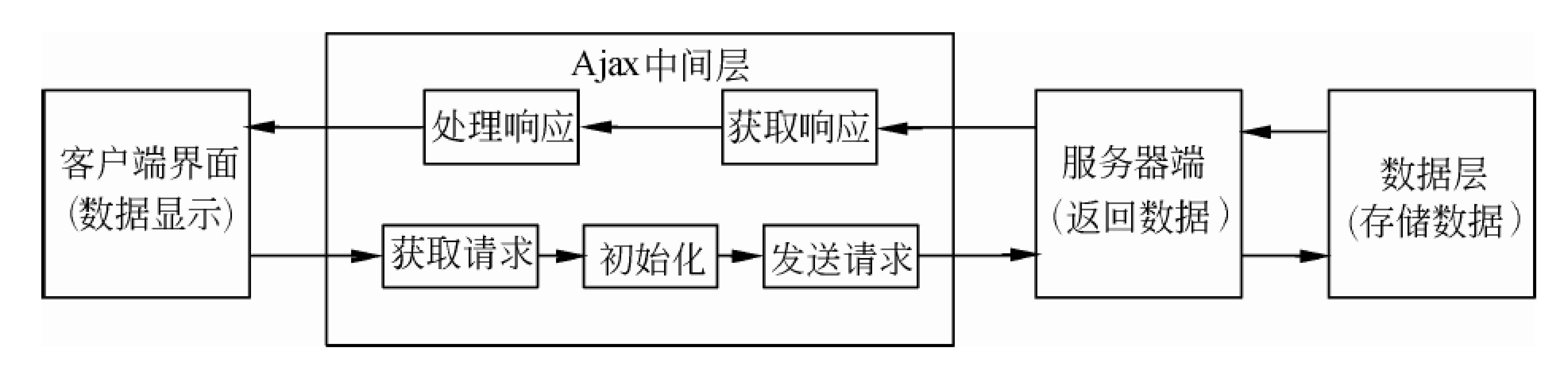


图 11-3　XMLHttpRequest 对象工作流程

在图 11-3 中，Ajax 中间层显示 XMLHttpRequest 对象的工作流程。当 Ajax 中间层从客户端界面获取请求信息之后，需要初始化 XMLHttpRequest 对象。初始化完成之后，通过 XMLHttpRequest 对象将请求发送给服务器端。服务器端获取请求信息后，处理并返回响应信息。然后 Ajax 中间层获取响应，通过 XMLHttpRequest 对象将响应信息和 Ajax 中间层所设置的样式信息进行组合，即处理响应。最后 Ajax 中间层将所有的信息发送给客户端界面，并显示由服务器返回的信息。

11.1.3　处理文本

前面详细介绍了 XMLHttpRequest 对象的内容，本节将通过一个具体的案例来讲解最简单的文本方法。

XMLHttpRequest 对象可以使用 GET 或 POST 方式发送请求，这与传统的 Web 编程是一样的。区别是，当使用 GET 发送请求时，必须把参数串追加到请求 URL 中；而使用 POST 时，则需要在调用 XMLHttpRequest 对象的 send()方法时发送参数串。

【实践案例 11-2】

下面创建一个发送消息的静态表单，然后使用 Ajax 的异步方式发送到服务器端，并最终输出消息列表。

（1）首先设计一个 HTML 页面 index.html，添加发送消息、提交按钮和结果显示区域，如下代码所示。

```
<h2>发送消息</h2>
 <table cellspacing="0" cellpadding="3">
     <tr>
       <th><label for="username">收件人：</label></th>
       <td>
         <input type="text" id="username" name="username" value=""
         style="width: 200px;"   />
         </td>
     </tr>
     <tr>
       <th style="vertical-align: top;"><label for="message">内容:
        </label></th>
       <td><textarea id="message" name="message" cols="40" rows="5"
        ></textarea></td>
     </tr>
     <tr>
       <th> </th>
       <td><input type="submit" name="pmsubmit btn" id="pmsubmit btn"
        value="发送"  onclick="sendMessage();" /></td>
     </tr>
   </table>
<h2>消息列表</h2>
```

```
<ul id="mlist">   </ul>
```

在上述代码中，单击“发送”按钮调用 sendMessage()函数以异步方式发送请求。

（2）如果要发送请求，必须先创建 XMLHttpRequest 对象，之后调用 XMLHttpRequest 对象的属性方法实现发送请求。这里使用的是 GET 方式提交到 Server.php 文件，因此参数会附加到 URL 上。具体实现代码如下所示：

```
<script type="text/Javascript">
var xmlHttp;
//省略创建 XMLHttpRequest 对象代码
function $(id){return document.getElementById(id);}
//发送 GET 请求
function sendMessage()
{
  //创建 XMLHttpRequest 对象
  createXMLHttpRequest();
  //定义一个变量保存输入的收件人
  var uname = encodeURIComponent($("username").value);      //收件人
  var mess = encodeURIComponent($("message").value);        //内容
  var para = "u="+uname+"&m="+mess;                        //组成参数字符串
  var url="server.php?"+para;                              //设置 URL 和参数
  xmlHttp.onreadystatechange=handleStateChange;            //指定回调函数
  xmlHttp.open("GET",url,true);                            //指定 GET 请求的数据
  xmlHttp.send(null);                                      //发送 GET 请求
}
</script>
```

在 sendMessage()函数中$("username").value 语句表示使用获取 id 为 username 的值，$("message").value 语句用于获取 id 为 message 的值。然后将获取的两个值组成一个新的字符串 para，每个属性之间用“&”符号隔开，属性和属性值用“=”进行赋值。

在 url 变量中将提交的服务器端文件与参数进行组合，它将作为 open()方法的第二个参数。onreadystatechange 属性设置处理服务器端响应的函数为 handleStateChange。最后调用 open()方法发送一个 GET 请求，并指定 URL，在这里 URL 中包含编码的参数。send()方法将请求发送给服务器。

（3）POST 与 GET 方式的不同之处在于 POST 允许发送任何格式、任何长度的数据，而 GET 方式最大只能发送 2KB 数据。下面代码实现用 POST 方式发送请求：

```
//发送 POST 请求
function sendMessage()
{
  //创建 XMLHttpRequest 对象
  createXMLHttpRequest();
  var uname = encodeURIComponent($("username").value);
  var mess = encodeURIComponent($("message").value);
  var para = "u="+uname+"&m="+mess;
```

```
    //定义一个变量保存 PHP 服务器端的文件名
    var url="server.php";
    xmlHttp.onreadystatechange=handleStateChange;     //指定回调函数
    xmlHttp.setRequestHeader("Content-Type","application/x-www-form-urlen
    coded;");
    xmlHttp.open("POST",url,true);                    //设置 POST 方式
    xmlHttp.send(para);                               //发送 POST 请求
}
```

为了确保服务器中知道有请求参数，需要调用 setRequestHeader()方法将 Content-Type 值设置为 application/x-www-form-urllencode。最后调用 send()方法并将数据作为参数传递给这个方法。

（4）无论使用 GET 方式，还是使用 POST 方式传递，其处理服务器端响应信息的回调函数相同。如下所示为 handleStateChange()函数的代码：

```
function handleStateChange()                    //处理结果的回调函数
{
    if((xmlHttp.readyState == 4)&&(xmlHttp.status == 200))
    {
        var result=xmlHttp.responseText;
        $("mlist").innerHTML+=result;
    }
}
```

（5）保存上面对 index.html 的修改。下面来创建服务器端页面 server.php，实现获取数据并输出结果的功能。具体代码如下所示：

```
<?php
header('Content-type:text/html;charset=utf-8');
if(isset($_GET['u'])&&isset($_GET['m']))
{
    $uname=$_GET['u'];
    $message=$_GET['m'];
    $create_at=date("Y-m-d H:i:s");
    echo "<li>我在".$create_at." 对‘".$uname."’说"."<br/>".$message.
    "<hr/></li>";
}
?>
```

（6）在浏览器中运行 index.html 页面，在表单中输入内容再单击“发送”按钮，之后会在下方显示结果，如图 11-4 所示。

在本案例第 4 步的回调函数中使用 JavaScript 的 innerHTML 属性将服务器返回的结果作为 HTML 格式显示到页面。与它对应的还有一个 innerText 属性，它可以将结果显示为普通文本格式，图 11-5 所示为此时的运行效果。



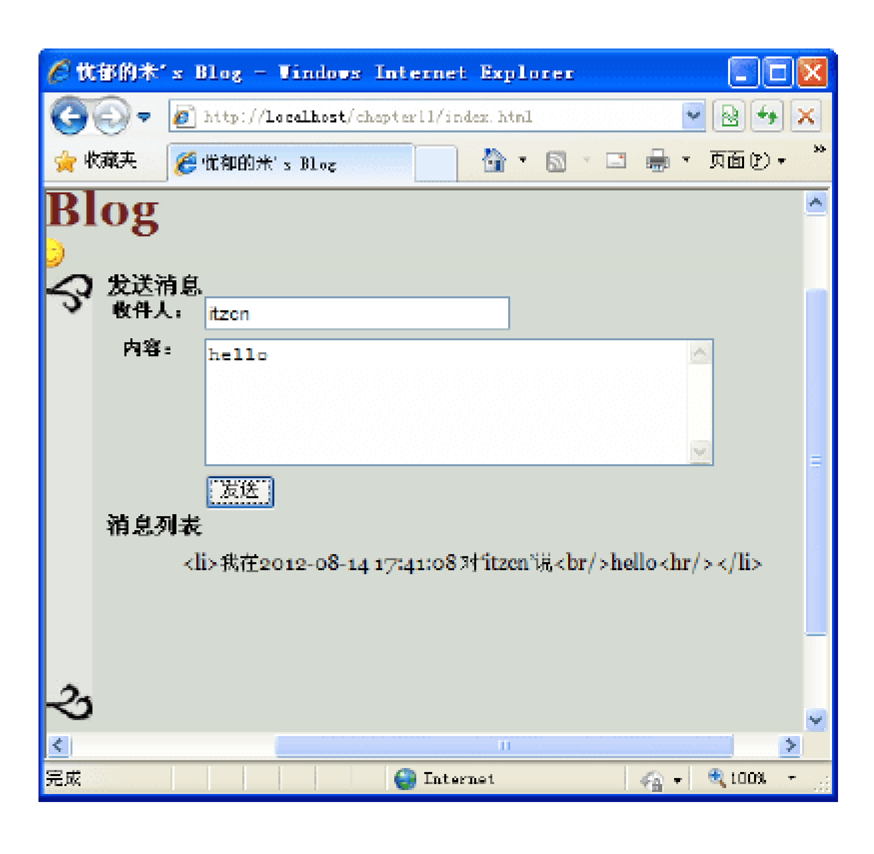


图 11-4　查看消息运行效果　　　　图 11-5　以文本方式显示结果

11.1.4　处理 XML

11.1.3 介绍了使用 XMLHttpRequest 对象发送 GET 和 POST 请求，然后处理服务器端返回的 HTML 文本。对于复杂结构的数据，在服务器端通常使用 XML 文件格式返回。此时 XML 数据的操作是重点，这些 XML 数据可以预先设定，也可以来自于数据库表或文件。

XMLHttpRequest 对象提供了一个 responseXML 属性专门用于接收 XML 响应。

【实践案例 11-3】

下面创建一个案例演示如何将服务器端返回的 XML 文件以列表形式显示到页面。具体步骤如下所示。

（1）首先创建服务器端的 PHP 页面 logs.php。作为示例我们直接输出一个 XML 格式的字符串，实际开发可能会用到从数据库中读取并输出，代码如下所示：

```
<?php
header("Content-Type: text/xml");
$xml=<<<XML
<?xml version="1.0" encoding="utf-8"?>\n
<logs>
<log><title>报道</title><user>ljq</user><s_content>网站恢复了，上来报道。</s_content><link>index.php?id=33</link></log>
<log><title>下雨啦</title><user>somboy</user><s_content>马上就要下雨啦...</s_content><link>index.php?id=4</link></log>
<log><title>问题求解</title><user>itzcn</user><s_content>怎么清除数据时保留库表中的部分数据</s_content><link>index.php?id=46</link></log>
<log><title>验证通过</title><user>itaod</user><s_content>邮箱验证通过了，高兴。。。</s_content><link>index.php?id=17</link></log>
</logs>
XML;
echo $xml;                    //输出上面定义的 XML 格式字符串
?>
```

（2）新建一个 list.html 文件作为客户端，在 body 的 onload 事件中调用 getAllLogs()函数，再添加结果显示区域，代码如下所示：

用 for 循环遍历节点中的元素，并且将遍历的元素读取到表格中。最后将表格中的数据显示在 id 为 logBody 的 div 标签中。

（5）在浏览器中运行 list.html，页面加载完成后会看到以表格形式显示 logs.php 文件中的数据，如图 11-6 所示。

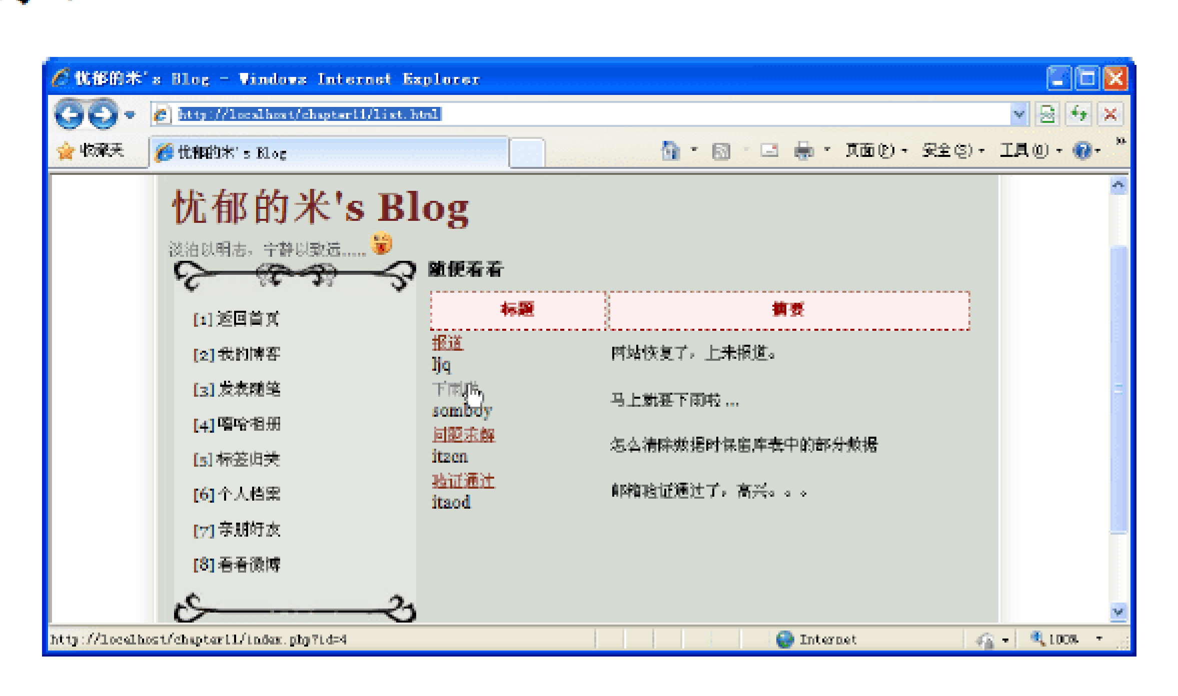
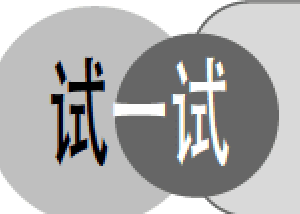


图 11-6　显示 XML 文件



目前出现了很多封装 XMLHttpRequest 对象的 Ajax 框架，像 jQuery、Extjs 和 Xajax 等，读者可以参考相关资料学习。

11.2　PHP 加密技术

对于一名优秀的 PHP 开发人员来说，除了在编写程序时要保证代码的高性能之外，还有一点是非常重要的，那就是程序的安全性保障。

使用加密技术可大大地增强安全性能。为此不仅可以使用 PHP 的内置加密函数，还可以使用功能强大的加密扩展库。下面将详细介绍这些内容。

11.2.1　内置加密函数

对于一些比较常用的加密算法，PHP 提供了非常有利的支持，像 md5、crypt 和 sha1，在需要时可以直接调用。

1．MD5 加密算法

MD5 全称是 Message-Digest Algorithm 5（信息-摘要算法），它主要用于让大容量信息在用数字签名软件签署私人密钥前被压缩成一种保密的格式，即把任何一个字符串变换成一定长的大整数。

在 PHP 中使用 MD5 加密算法非常方便，只需调用 md5()函数即可。该函数语法如下所示：

```
string md5(string $str [, bool $raw_output ])
```

```
<?php
$str = "hello";
echo crypt($str);                //输出$1$/4...I4.$8VOJAsu0g/KtcLWf0bHI90
echo crypt($str,"000");          //输出 00nI9k6eXQgGU
?>
```

11.2.2 加密扩展

除了前面介绍的内置加密函数外，PHP 还提供了非常强大的加密扩展函数库。这些库可以使数据更加安全，其中有提供双向加密的，还有为大量散列算法提供的，如表 11-4 所示。

表 11-4 PHP 加密扩展库

扩展包和函数库	详细描述
Mcrypt	提供广泛的加密功能，可以双向加密，用来加密大型文件或数据流
Mhash	支持最流行的散列算法，如 MD5
Crypt_Blowfish	可以进行快速双向加密，可以选择使用或不使用密钥
Crypt_RSA	提供 RSA 一样的密钥生成、加密/解密、数位签署及签署验证功能
Crypt_HMAC	这个类可用来计算兼容 RFC 2104 的散列值
Crypt_DiffieHellman	这是一个在 PHP 5 上的 Diffie-Hellman 密钥交换协议

PHP 提供了众多的加密扩展，但比较常用的就两三种，下面将对 PHP 常用的库进行详细的讲解。

1. Mcrypt 库

Mcrypt 库中有大量的加密函数来对文件和数据流进行加密，Mcrypt 库的加密都是针对固定长度的数据块，一般为 64 或 128 字节。但由于明文的长度是不固定的，再加上使用相同的密钥来加密相同的明文，可能会得到相同的结果。

Mcrypt 库支持多种区块的加密算法，包括 Blowfish、DES、TripleDES、SAFER-SK128、TWOFISH、TEA、RC2、3-WAY、SAFER-SK64 和表 11-5 所示的 6 种加密模式。

表 11-5 加密模式

模式缩写	使用例子	详细信息
CBC	MCRYPT_MODE_CBC	密文块链模式，用来加密文件
CFB	MCRYPT_MODE_CFB	密文反馈模式，建议用来加密字节流
STREAM	MCRYPT_MODE_STREAM	特色的流模式，需要资料流算法时使用，例如 WAKE 或 RC4
ECB	MCRYPT_MODE_ECB	电子密码本模式，适合随机数据，你可以用这种模式来加密不同的密码
OFB	MCRYPT_MODE_OFB	8 位输出反馈模式，专门用在不容许出错的应用系统
NOFB	MCRYPT_MODE_NOFB	N 位输出反馈模式，兼容 OFB，但更加安全

Mcrypt 库提供了 30 多个加密和解密函数，这里不一一罗列，仅介绍最重要、最常用的两个函数：mcrypt_encrypt()和 mcrypt_decrypt()。

❑ mcrypt_encrypt()函数

mcrypt_encrypt()是 Mcrypt 库中的一个加密函数，其语法如下：

```
    string  mcrypt encrypt(string  $dipher,string  $key  ,string  $data,string
$mode[,string $iv]);
```

其中，$dipher 是要使用的加密算法，$key 是使用的加密密钥，$mode 是指定的 6 个加密模式中的一个加密模式，$data 是要加密的数据，而参数$vi 是对 cbc、cfb、ofb 以及流模式中使用的某种算法。该函数执行后返回加密后的结果字符串。

例如，下面的代码使用 DES 的加密算法对“hello”进行加密，并输出结果。

```
    <?php
    $str = "hello";              //加密内容
    $key = "111";                //密钥
    $cipher = MCRYPT_DES;        //密码类型
    $modes = MCRYPT MODE CBC;    //密码模式
    $iv=mcrypt_create_iv(mcrypt_get_iv_size($cipher,$modes),MCRYPT_RAND);
//初始化向量
    echo "原文: " . $str . "<p>";
    $str_encrypt = mcrypt_encrypt ( $cipher, $key, $str, $modes, $iv ); // 加
密函数
    echo "密文: " . bin2hex($str_encrypt) . " <p>";
    ?>
```

运行结果如下：

```
    原文: hello
    密文: cecf2afd3592303a
```

❑ mcrypt_decrypt()函数

mcrypt_decrypt()是针对 mcrypt_encrypt()函数进行解密的函数，其语法如下：

```
    string  mcrypt decrypt(string  $dipher,string  $key  ,string  $data,string
$mode[,string $iv]);
```

其中各个参数含义与 mcrypt_encrypt()函数相同。例如，下面的示例代码对使用 mcrypt_encrypt()函数加密的$str_encrypt 进行解密。

```
    <?php
    $str decrypt = mcrypt decrypt ( $cipher, $key, $str encrypt, $modes, $iv );
// 解密函数
    echo "还原: " . $str_decrypt;        //还原: hello
    ?>
```

2. Mhash 库

Mhash 是一个免费的加密函数开源库，它为 PHP 加密的大量散列值算法提供一个有用

的接口。可以用这些算法校验信息和进行数据验证等，还可以处理密码。

Mhash 库中最常用的是 mhash()函数，其语法格式如下：

```
string mhash(int $hash,string $data[,string $key]);
```

$hash 参数用于设置散列算法，表 11-6 列出了该函数支持的算法；$data 参数是要进行散列的密码。

表 11-6　Mhash 支持的散列算法

MHASH_ADLER32	MHASH_HAVAL128	MHASH_MD4	MHASH_SHA256
MHASH_CRC32	MHASH_HAVAL160	MHASH_MD5	MHASH_TIGER
MHASH_CRC32B	MHASH_HAVAL192	MHASH_RIPEMD160	MHASH_TIGER128
MHASH_GOST	MHASH_HAVAL256	MHASH_SHA1	MHASH_TIGER160

例如，如下代码使用 mhash()函数和 MD5 算法对“hello”进行加密：

```
<?php
$str = "hello";
$PASSMHASH = mhash(MHASH MD5,$str);
echo "密码散列值: ".bin2hex($PASSMHASH); //密码散列值: 2a95f3d7706cbeafc
8f745e07d53a084
?>
```

3. PEAR 的 Crypt_RSA PEAR 库

Crypt_RSA PEAR 库允许开发人员使用任意长度的密钥来加密数据，它以 RSA 区块加密技术为基础，支持双向加密，同时支持任意长度的密钥来加密和解密。

由于 Crypt_RSA 需要执行大量的数学计算，因此需要借助一些其他扩展才能使用，如表 11-7 所示。

表 11-7　计算扩展模块

模块名称	支持版本
PECL big_int 扩展模块	需要 PEAR 包 1.0.3 以上版本
PHP GMP 扩展模块	PHP 版本
PHP BCMath 扩展模块	PHP4/PHP5

如下代码演示了如何使用 Crypt_RSA PEAR 库对数据进行加密：

```
<?php
    require_once 'Crypt/RSA.php';
   //产生一双对称密钥
   function generate key pair(){
      global $public key,$private key;
      $key pair = new Crypt RSA KeyPair(32);
      // 从这双密钥中提取公钥
      $public_key = $key_pair->getPublicKey();
      // 从这双密钥中提取密钥
```

```
        $private key = $key pair->getPrivateKey();
    }
    $file = 'textfile.txt';
    generate_key_pair();
    $plain_text = file_get_contents($file);
    // 把公钥表达为一个字符串
    $key = Crypt RSA Key::fromString($public key->toString());
    $rsa_obj = new Crypt_RSA;
    // 用密钥 $key 加密 $plain_text
    $encrypted = $rsa obj->encrypt($plain text, $key);
    $encrypted_file = @fopen('encrypted.txt','w');
    $ok_encrypt = fwrite($encrypted_file,$encrypted);
?>
```

4. Crypt_HMAC 库

Crypt_HMAC 库提供了一个类可以用来计算 RFC2104 兼容的散列值。该类的使用方法非常简单，只需指定密钥和明文即可，而且还支持 MD5 和 SHA-1 算法。语法如下所示：

```
    $var = new Crypt_HMAC(string $key, string $dipher);
```

其中$key 是要提供的密钥，$dipher 是要使用的散列算法。如下代码演示了如何使用 Crypt_HMAC 库对数据进行加密：

```
<?php
    require once 'HMAC.php';
    // 把字符"defre" 重复 40 次来产生一个密钥
    $key = str repeat(chr(defre), 40);
    // 产生一个 Crypt_HMAC 类的实体
    $crypt = new Hmac($key, 'md5');
    // 散列函数
    echo $crypt->sha1('Hello');
    $key = str_repeat(chr(0xcc), 10);
    $data = str_repeat(chr(0xdd), 50);

    // 把散列函数的密钥设定为 $key
    $crypt->setKey($key);
    echo $crypt->sha1($data)."\n";
?>
```

11.3 PHP 开发编程规范

俗话说“没有规矩不成方圆”，开发程序亦是如此，良好的编程风格和规范对于开发人员及项目管理者都是非常重要的。假设你要维护的一个项目没有文档、代码杂乱无章，甚至连程序注释和缩进也没有。这对你来说肯定是一个噩梦。

本节将介绍一些开发中需要遵守的规范以及一些好的编程习惯，以此帮助读者成为一名优秀的开发人员。

11.3.1 包含文件

对于多个页面都需要使用的功能，将它封装成函数，再提取出来具有通用函数的包含文件，文件后缀以.inc 来命名，表明这是一个包含文件。

如果有多个.inc 文件需要包含多页面，可把所有.inc 文件封装在一个文件里面，具体到页面只需要包含一个.inc 文件就可以了。例如：

```
xxx_session.inc
xxx_comm..inc
xxx setting.inc
mysql_db.inc
```

把以上文件以以下方式，封装在 xxx.basic.inc 文件里面。然后用如下一行语句进行包含：

```
require_once("xxx_basic.inc");
```

这里注意，是否需要封装到一个文件应该视情况而定，如果每个 inc 的功能是分散到不同的页面使用的话就不建议封装。

另外，一般包含文件不需要直接暴露给用户，所以应该放在 Web Server 访问不到的目录，避免因为配置问题而泄露设置信息。

11.3.2 命名规范

虽然开发人员喜欢按照自己的习惯和风格来编写代码，但是在编码中，代码的格式、布局、命名方式以及语句规范等应该遵循一定的规范。这些规定虽然并非强制性的，但为了加强代码结构的逻辑性和易读性，遵循它们是非常必要的。下面主要介绍的是 PHP 中与命名有关的规范。

1. 普通变量

普通变量命名遵循以下规则。

- **所有字母都使用小写。**
- **对于一个变量使用多个单词的，使用“_”作为每个词的间隔。**

例如：$file_name、$output_product_price 等。

2. 静态变量

静态变量命名遵循以下规则。

- **静态变量使用小写的 s_开头。**
- **静态变量所有字母都使用小写。**
- **多个单词组成的变量名使用“_”作为每个词的间隔。**

例如：$s_file_name、$s_output_product_price 等。

3．局部变量

局部变量命名遵循以下规则。

- 所有字母使用小写。
- 变量使用“_”开头。
- 多个单词组成的局部变量名使用“_”作为每个词间的间隔。

例如：$_file_file、$_output_product_price 等。

4．全局变量

全局变量应该带前缀“g”,知道一个变量的作用域是非常重要的。例如：

```
global $gLOG LEVEL;
global $gLOG_PATH;
```

5．全局常量

全局变量命名遵循以下规则。

- 所有字母使用大写。
- 全局变量多个单词间使用“_”作为间隔。

例如：$FILE_NAME、$OUTPUT_PRODUCT_PRICE 等。

6．Session 变量

Session 变量命名遵循以下规则。

- 所有字母使用大写。
- Session 变量名使用 S_开头。
- 多个单词间使用“_”间隔。

例如：$S_FILE_NAME、$S_OUTPUT_PRODUCT_PRICE 等。

7．类

PHP 中类命名遵循以下规则。

- 以大写字母开头。
- 多个单词组成的变量名，单词之间不用间隔，各个单词首字母大写。

例如：class MyClass 或 class DbOracle 等。

8．方法或函数

方法或函数命名遵循以下规则。

- 首字母小写。
- 多个单词间不使用间隔，除第一个单词外，其他单词首字母大写。

例如：function myFunction()或 function myDbOracle()等。

9．缩写词

当变量名或者其他命名中遇到缩写词时，参照具体的命名规则，而不采用缩写词原来的全部大写的方式。

例如：

```
function myPear (不是 myPEAR)
functio getHtmlSource (不是 getHTMLSource)。
```

11.3.3　代码编写规范

一个好的代码编写风格和规范对于一个项目来说非常重要，这包含使用缩进、结构控制和注释等方面。

1．代码缩进

在编写代码的时候，必须注意代码的缩进规则，我们规定代码缩进规则是：使用 4 个空格作为缩进。

例如：

```
for ( $i=0;$i<6;$i++ )
{
    echo $i;
}
```

2．大括号{ }书写规则

在程序中进行结构控制代码编写，如 if、for、while、switch 等结构，大括号传统的有如下两种书写习惯。

（1）左大括号{直接跟在控制语句之后不换行。例如：

```
for ($i=0;$i<6;$i++) {
    echo $i;
}
```

（2）左大括号{在控制语句下一行。例如：

```
for($i=0;$i<6;$i++)
{
    echo $i;
}
```

从实际书写来讲这两种都不影响程序的规范和用 phpdoc 生成文档，所以可以根据个人习惯来采用上面的两种方式。但是要求在同一个程序中只使用其中一种，以免造成阅读的不方便。

3．小括号( )和函数、关键词等

小括号、关键词和函数遵循以下规则。

- 不要把小括号和关键词紧贴在一起，要用一个空格间隔，例如 if ($a<$b)。
- 小括号和函数名之间没有空格，例如$test = date("ymdhis")。
- 除非必要，否则不要在 return 返回语句中使用小括号，例如 return $a。

4．=符号书写

在程序中“=”符号的书写遵循以下规则。

- 在=符号的两侧，均需留出一个空格，例如$a = $b、if ($a == $b)。
- 在一个声明块或者实现同样功能的一个块中，要求“=”号尽量上下对齐，左边可以为了保持对齐使用多个空格，而右边要求空一个空格。

例如：

```
$testa   = $aaa;
$testaa  = $bbb;
$testaaa = $ccc;
```

5．if else、swicth、for、while 等语句书写

对于控制语句的书写遵循以下规则。

- 在 if 条件判断中，如果用到常量判断条件，将常量放在等号或不等号的左边。

例如：

```
if ( 6 == $errorNum )
```

因为如果在等式中漏了一个等号，语法检查器将会提示错误，从而可以很快找到错误位置，这样的写法要多注意。

- switch 结构中必须要有 default 块。
- 在 for 和 while 的循环使用中，要警惕 continue、break 的使用，避免产生类似 goto 的问题。

6．类的构造函数

如果要在类里面编写构造函数，必须遵循以下规则。

- 不能在构造函数中有太多实际操作，最多初始化一些值和变量。
- 不能在构造函数中因为使用操作而返回 false 或者错误，因为在声明和实例化一个对象的时候，是不能返回错误的。

7．语句断行

在代码书写中遵循以下原则。

- 尽量保证程序语句一行就是一句，而不要让一行语句太长而换行。

- 尽量不要使一行的代码太长，一般控制在 80 个字符以内。
- 如果一行代码太长，应使用类似于“=”的方式断行书写。
- 对于执行数据库的 SQL 语句操作，尽量不要直接写在函数内，而先用变量定义 SQL 语句，然后在执行操作的函数中调用定义的变量。

例子：

```
$sql = "SELECT uid,username,password,address,age,postcode FROM test t ";
$sql .= " WHERE username='abc' and age>24";
$res = mysql_query($sql);
```

8. 不要直接使用数字

在源代码中直接使用的数字被认为是不可思议（赤裸裸）的数字，因为包括作者在内，过段时间再看代码时，没人能看懂它的含义。例如：

```
if (22 == $foo)
{
    start thermo nuclear war();
}
elseif (19 == $foo)
{
    refund_lotso_money();
}
```

解决方法：应该用 define()来给想表示某样东西的数值一个真正的名字，而不是直接采用数字。例如：

```
define("PRESIDENT WENT CRAZY", "22");
define("WE GOOFED", "19");
if (PRESIDENT WENT CRAZY == $foo)
{
     start thermo nuclear war();
}
elseif (WE_GOOFED == $foo)
{
    refund lotso money();
}
```

9. 判断 true 和 false

在判断 true 和 false、0 和 1 时要遵循以下规则。

- 不能使用 0/1 代替 true/false，在 PHP 中这是不相等的。
- 不要使用非零的表达式、变量或者方法直接进行 true/false 判断，而必须使用严格的完整 true/false 判断。
- 不使用 if ($a)或者 if (check())，而使用 if (FALSE != $a)或者 if (FALSE != check())。

11.3.4 程序注释

每个程序均必须提供必要的注释，书写注释要求规范为利用 phpdoc 生成 PHP 文档做准备。程序注释的原则如下。

- **注释中除了文件头的注释块外，其他地方都不使用//注释，而使用/* */的注释。**
- **注释内容必须写在被注释对象的前面，不写在一行或者后面。**

1. 程序头注释块

每个程序头部必须有统一的注释块，规则如下。

- **必须包含本程序的描述。**
- **必须包含作者。**
- **必须包含书写日期。**
- **必须包含版本信息。**
- **必须包含项目名称。**
- **必须包含文件的名称。**
- **重要的使用说明，如类的调用方法、注意事项等。**

例如，下面的参考代码：

```
// +----------------------------------------------------------------------+
// | PHP version 4.0                                                      |
// +----------------------------------------------------------------------+
// | Copyright (c) 1997-2001 The PHP Group                                |
// +----------------------------------------------------------------------+
// | This source file is subject to  of the PHP license,                  |
// | that is bundled with this packafile LICENSE, and is                  |
// | available at through the world-web at                                |
// | http://www.php.net/license/2 02.txt.                                 |
// +----------------------------------------------------------------------+
// | Authors: Stig Bakken <ssb@fast.no>                                   |
// |          Tomas V.V.Cox <cox@idecnet.com>                             |
// +----------------------------------------------------------------------+
// $Id: Common.php,v 1.8.2.3 2001/11/13 01:26:48 ssb Exp $
```

2. 类的注释

类的注释可以参考下面的示例代码：

```
/**
* @ Purpose: 访问数据库的类，以 ODBC 作为通用访问接口
* @Package Name: Database
* @Author: Forrest Gump gump@crtvu.edu.cn
* @Modifications: No20020523-100:
```

```
* odbc_fetch_into()参数位置第二和第三个位置调换
* @See: (参照)
*/
class Database
{
    …
}
```


3. 函数和方法的注释

函数和方法的注释写在函数和方法的前面，采用类似于下面例子的规则：

```
/**
* @Purpose: 执行一次查询
* @Method Name: Query()
* @Param: string $queryStr SQL 查询字符串
* @Param: string $username 用户名
* @Author: Lee
* @Return: mixed 查询返回值（结果集对象）
*/
function Query($queryStr, $username)
{…}
```

4. 变量或者语句注释

程序中变量或者语句的注释遵循以下原则。

- 写在变量或者语句的前面一行，而不写在同行或者后面。
- 注释采用/* */的方式。
- 每个函数前面要包含一个注释块，内容包括函数功能简述、输入/输出参数、预期的返回值、出错代码定义。
- 注释完整、规范。
- 把已经注释掉的代码删除，或者注明这些已经注释掉的代码仍然保留在源码中的特殊原因。

参考下面的示例代码：

```
/**
* @Purpose: 数据库连接用户名
* @Attribute/Variable Name: db_user_name
* @Type: string
*/
var db_user_name;
```

11.3.5 项目结构规范

项目中的程序文件名和目录名均应该采用有意义的英文方式命名，而不使用拼音或无

意义的字母，同时均必须使用小写字母，多个词间使用下划线“_”间隔。

在开发独立的 PHP 项目时使用规范的文件目录结构，这有助于提高项目的逻辑结构合理性，对扩展以及团队开发均有好处。

一个完整独立的 PHP 项目通常的文件和目录结构如下：

```
/ 项目根目录
/manage 后台管理文件存放目录
/css CSS 文件存放目录
/doc 存放项目文档
/images 所有图片文件存放路径（在里面根据目录结构设立子目录）
/scripts 客户端 JavaScript 脚本存放目录
/tpl 网站所有 HTML 的模版文件存放目录
/error.php 错误处理文件（可以定义到 apache 的错误处理中）
```

以上目录结构是通常的目录结构，根据具体应用的具体情况可以考虑不用完全遵循，但是尽量做到规范化。

在实现方式上，如果是对性能要求不是很高的项目和应用，建议采用 PHP 和 HTML 代码分离的方式，即采用模板的方式处理，这样对程序逻辑结构更加清晰有利，也有助于开发过程中人员的分工安排，同时还可对日后项目的页面升级改版提供更多便利。

在实现架构上，尽量采用 OOP 的思想开发，尤其在 PHP 5 以后，对于面向对象的开发功能大大提高。我们建议将独立的功能模块尽量写成函数调用，对于一整块业务逻辑，建议封装成类，既可以提高代码可读性，也可以提高代码重用性。例如，通常将对数据库的接口封装成数据库类，这有利于平台的移植。

11.4 习题

一、填空题

（1）假设要创建一个运行在 IE 浏览器上的 XMLHttpRequest 对象应该使用代码____________。

（2）XMLHttpRequest 对象的____________属性表示 HTTP 响应的数字状态码。

（3）如下所示为一段回调函数的代码，在空白处填写代码使 xml 变量中保存的是返回的 XML 结果。

```
function handleStateChange()                    //处理结果的回调函数
{
    if((xmlHttp.readyState == 4)&&(xmlHttp.status == 200))
    {
        var result=             ;
    }
}
```

（4）假设要使用 MD5 对字符串“PHP”进行加密，应该使用代码____________。

二、选择题

（1）下面关于 Ajax 的描述不正确的是____________。

A．Ajax 不是一种新技术

B．Ajax 与服务器通信时不刷新页面

C．Ajax 可以减轻服务器和带宽的负担

D．Ajax 可以持续保持一个 HTTP 请求

（2）下面代码中能判断服务器端有响应的是____________。

A．(xmlHttp.readyState == 4)&&(xmlHttp.status == 200)

B．(xmlHttp.readyState == 1)&&(xmlHttp.status == 100)

C．(xmlHttp.readyState == 0)&&(xmlHttp.status == 200)

D．(xmlHttp.readyState == 'ok')&&(xmlHttp.status == 'ok')

（3）下面不属于 PHP 内置加密函数的是____________。

A．md5()

B．sha1()

C．crypt()

D．mcrypt()

（4）下面属于 Mcrypt 库中解密函数的是__________。

A．md5()

B．mcrypt_encrypt()

C．mcrypt_decrypt()

D．sha1()

（5）下面不属于 Mhash 函数支持的散列算法的是__________。

A．MHASH_MD4

B．MHASH_MD5

C．MHASH_RIPEMD160

D．MHASH_SJI

三、上机练习

1．使用 Ajax 发送 GET 和 POST 请求

假设在 HTML 页面中制作了一个用于输入视频信息的表单，代码如下所示：

```
<form action="#">
  视频标题：<input type="text" id="title"/><br/>
  视频标签：<input type="text" id="tag"/><br/>
  视频长度：<input type="text" id="length"/><br/>
  所属类别：<input type="text" id="classes" /><br/>
  所需积分：<input type="text" id="score"/><br/>
```

```
        <input type="button" value="GET 方式提交" onclick="RequestGet()"/>
        <input type="button" value="POST 方式提交" onclick="RequestPOST()"/>
    </form> <br/>
    <div id="message" ></div>
```


现在要求读者完善程序包括客户端 GET 和 POST 请求的提示，以及服务端的响应代码。

2．实现 Ajax 的会员注册功能

本次上机要求读者使用 Ajax 实现一个会员注册功能：用户注册的时候要填写登录名称，当光标离开“登录名”文本框时对它进行检测，判断该名称是否已经被使用。如果是，则在页面不刷新的情况下给出相应的提示，如图 11-7 所示。如果可用也可给出提示，如图 11-8 所示。



图 11-7　不可用时的效果



图 11-8　可用时的效果

11.5　实践疑难解答

11.5.1　如何解决 PHP 接收的参数是乱码问题



如何解决 PHP 接收的参数是乱码问题

网络课堂：http://bbs.itzcn.com/thread-19700-1-1.html

【问题描述】：大家好，刚接触 Ajax 做了一个小练习就出问题了，具体症状是：做了一个输入表单用 Ajax 方式提交到 PHP，然后发现在接收的时候不是获取不到，就是获取乱码。

【解决办法】：这个问题很让人头疼，也是开发人员经常遇到的问题。主要是因为使用 Ajax 传送和接收中文参数时，如果不在客户端和服务器做相应的处理就会出现乱码问题。本人总结了两种最常用的解决方案供你参考。

1. 客户端乱码

第一种是向服务器端发送的参数有中文，在服务器端接收参数值时发生乱码。例如：

```
var url="search.php?key=窗内网";
xmlHTTP.open ("post",url,true);
```

解决方法：利用 JavaScript 提供的 escape()、encodeURIComponent()或 encodeURI()方法在客户端对参数进行编码，它们的作用如下。

（1）传递参数时需要使用 encodeURIComponent，这样组合的 URL 才不会被#等特殊字符截断。例如：

```
<script language="javascript">document.write('<a href="http://passport.
baidu.com/?logout&aid=7&u='+encodeURIComponent("http://cang.baidu.com/bruce
42")+'">退出</a>;');</script>
```

（2）进行 URL 跳转时可以整体使用 encodeURI，使用这个方法编码的字符在 PHP 中可以使用 urldecode()函数解码。例如：

```
location.href=encodeURI(http://cang.baidu.com/do/s?word=特价&ct=21);
```

（3）JavaScript 使用数据时可以使用 escape。escape 对 0～255 以外的 Unicode 值进行编码时输出%u****格式，其他情况下与 encodeURI 和 encodeURIComponent 编码结果相同。

2. 服务器端乱码

第二种就是服务器端向客户端输出中文时出现乱码，也就是客户端用 responseText 或 responseXML 取值时包含的中文是乱码。

原因：Ajax 在接收 responseText 或 responseXML 的值时是按照 UTF-8 的格式来解码的。所以，如果服务器端发送的数据不是 UTF-8 的格式，那么接收 responseText 或 responseXML 的值有可能为乱码。

解决办法：在服务器指定发送数据的格式。在 PHP 文件中用下面一行语句：

```
header("content-type:text/html;charset= UTF-8");   //返回的是 TXT 文本文件
```

或者：

```
header("content-type:text/xml;charset=UTF-8");    //返回的是 XML 文件
```

11.5.2 关于会员注册时密码加密的问题



关于会员注册时密码加密的问题

网络课堂：http://bbs.itzcn.com/thread-19701-1-1.html

【问题描述】：大家看看这个注册页面 register.php。

```
<form name="form1" action="r.php" method="post">
```

```
        昵称:<input type="text" name="user" maxlength="20" size="20" /><br /><b/>
        密码: <input type="password" name="pwd" maxlength="20" size="20" /><br
/><br />
         <input type="submit" name="submit" value="login" />
    </form>
```


假如说在此页面进行 md5 加密再传到 r.php，那密码还是有可能被截获啊。请大家给讲讲安全方面具体该怎么做?

【解决办法】: 这个是 Web 天生的局限性，在浏览器上的数据被截获是无法避免的，除非用 SSL 加密，但是可以最大程度避免这种情况，做法如下。

（1）数据正常提交至 r.php。

（2）获取一个当前时间 time()。

（3）将密码组合 time()后 md5。

（4）将组合后的密码 md5 值和之前获取的 time()一起存入数据库。

当以后需要登录时首先获取用户名对应的 time()，再组合上密码进行 md5 和数据库内的密码 md5 对比。

第 12 章 相册管理系统

随着 Web 2.0 的兴起，越来越多的人热衷于在网络社区上交流和分享。与此同时也出现了很多 Web 2.0 的应用，像圈子、博客和个人空间等。在这些应用中都少不了相册，它提供了个性的图片展示以及可以在线分享和浏览。

本章以相册为主题，并使用 PHP 和 MySQL 来实现。从最初的功能分析和数据库设计开始，再到搭建系统架构和公共模块，然后实现查看相册、查看图片、上/下一张、管理相册以及上传图片等功能。

本章学习要点：

- 了解相册系统的功能划分和数据库设计
- 熟悉相册的运行过程
- 掌握显示相册列表的方法
- 掌握查看图片详细信息的方法
- 熟悉相册后台功能的实现过程
- 熟悉系统中控制器与视图的交互过程

12.1 系统分析

俗话说“知己知彼，百战不殆”，在开发一个系统之前首先需要对系统进行透彻的分析。有一个明确的需求分析之后，再确定系统的结构如何运行、实现哪些功能、数据库如何设计以及具体模块和页面的制作。

12.1.1 功能分析

相册的主要功能无非是对照片的浏览与查看，但是为了方便浏览，可以对照片按相册进行保存。当有了新照片时可以通过上传添加到相册中，还可以对现有的照片进行删除和更新。

相册系统主要需要提供以下的功能。

- 查看所有相册　首页会在列表中显示所有的相册和封面图片。
- 查看照片　按相册查看包含的照片列表。
- 查看图片大图　主要是指查看图片的原始大小，在这里可以滚动查看上一张和下一张。
- 管理员登录　通过登录验证是否具有管理权限，从而增加相册的安全性。

- **上传照片** 这是相册的主要功能之一。
- **创建相册** 照片必须存放到相册中才能显示。
- **相册和图片管理** 主要是指对相册和图片的信息进行更新和删除。
- **图片的快速操作** 像将图片设置为封面、复制图片地址以及移动图片到相册等。

根据上面对相册特点的介绍和功能的分析，在本系统中主要将相册分为前台、管理员和后台 3 个模块。图 12-1 所示为相册的模块设计图。

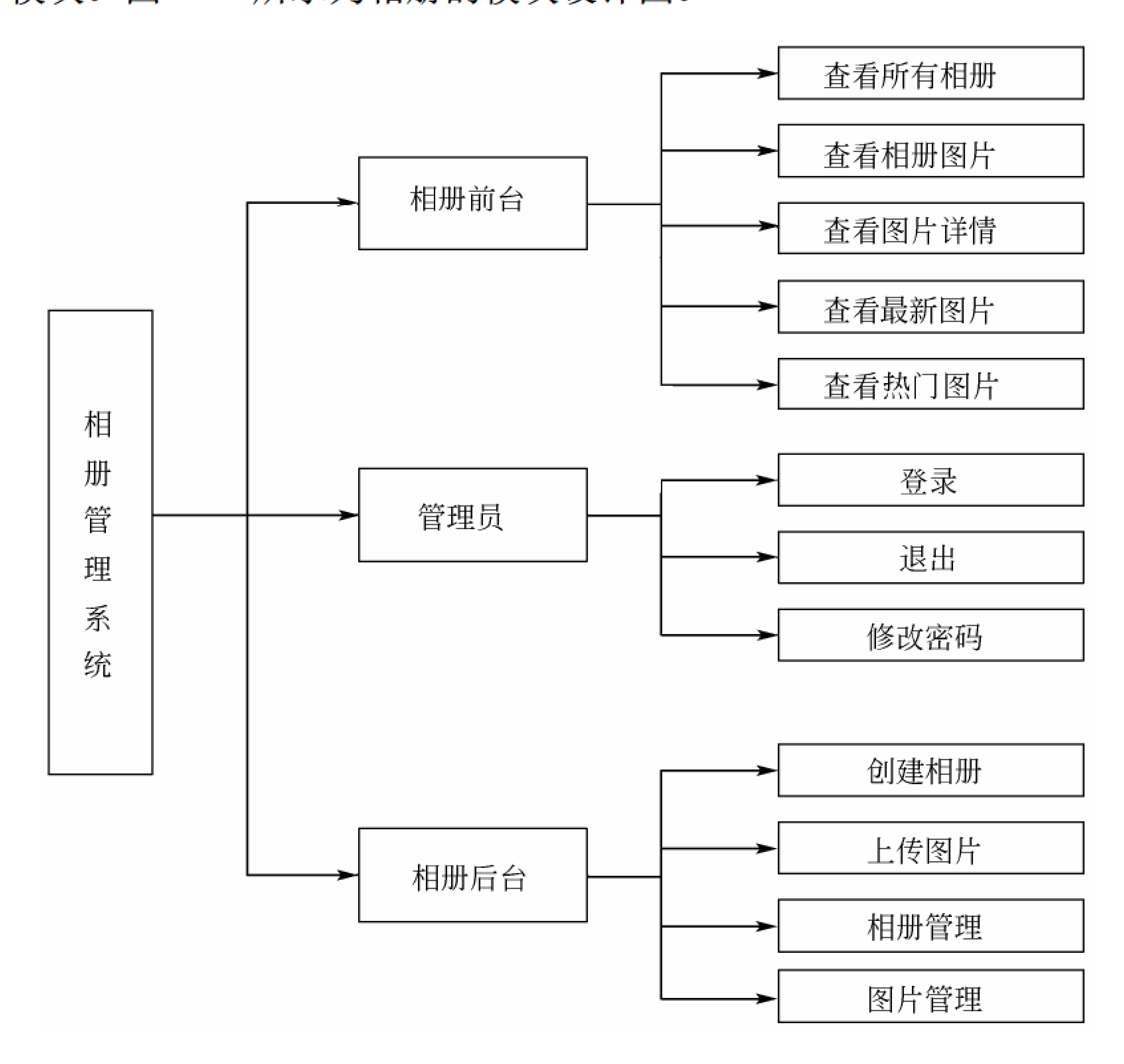


图 12-1　系统功能模块设计图

12.1.2　数据库设计

经过功能分析，划分了相册系统的功能模块，接下来对系统的数据库进行设计。这包括考虑系统实现都需要哪些数据表、数据表中包括哪些字段、这些字段用来做什么等。下面将对系统中使用到的数据表及字段详细说明。

本系统采用 MySQL 数据库，建立一个名为 db_gallery 的实例数据库，下面详细介绍包含表的信息。

1. 管理员表

管理员表（my_admin）保存了相册中具有管理权限的用户名和密码，各字段详细结构如表 12-1 所示。

表 12-1 my_admin 表字段信息

字段	数据类型	是否为空	备注
id	int	否	主键，自动增长
username	varchar(50)	否	管理员用户名
userpass	varchar(50)	否	管理员密码
create_time	int	否	创建时间

2．相册表

相册表（my_albums）主要用于存储照片分类的名称、封面图片、创建时间和描述等信息。表 12-2 列出了该表的字段详细结构。

表 12-2 my_albums 表字段信息

字段	数据类型	是否为空	备注
id	int	否	主键、自增，分类编号
name	varchar(200)	否	相册名称
cover	int(11)	否	相册封面图片
create_time	int(11)	否	创建时间
private	tinnyint	否	是否公开
desc	text	否	相册的描述

3．照片表

照片表（my_imgs）主要用于存储照片名称、简介、浏览次数和添加日期等。表 12-3 列出了该表的字段详细结构。

表 12-3 my_imgs 表字段信息

字段	数据类型	是否为空	备注
id	int	否	主键、自增，图片编号
album	smallint(4)	否	相册编号
name	varchar(100)	否	图片名称
dir	varchar(10)	否	保存路径
pickey	varchar(32)	否	图片标识
ext	varchar(10)	否	扩展名
status	tinnyint	否	状态
hits	int(11)	否	浏览次数
create_time	int(11)	否	创建时间
private	tinnyint	否	是否私有
private_pass	varchar(50)	否	私有访问密码
author	int(11)	否	上传用户的 id
desc	text	否	图片描述

12.2 公共模块

在确定系统的功能模块以及数据库的设计之后，下面从公共模块的设计开始讲解、逐

步实现各个功能。对于一个系统来说，很多功能或者模块是相同的。因此，可以将这些代码封装成公共模块方便以后的调用。

12.2.1　搭建项目架构

在对相册系统的功能进行了划分，并设计了所需的数据库后，接下来开始实现系统，实现的第一步是为系统选择一种架构或者设计模式（实现方式）。

对于简单的小型系统可以直接将 PHP 脚本混合到 HTML 代码中，这种方式的缺点是维护和扩展比较难、代码冗余。为此，实际开发时通常会选择一个 PHP 框架，像 Zend 或者 thinkPHP 等，框架可以将显示与逻辑相分离，但是需要增加对框架学习的成本。

本系统在设计时采用了基于 MVC 的设计模式。MVC 要求对项目分层（模型层 Model、视图层 View 和控制层 Controller），虽然要花费额外的工作时间，但项目的结构清晰，项目的应用通过模型可以得到更好的体现，这对需求变更和后期维护提供了很好的支持。

在 Apache 下新建一个名为 mygallery 的目录作为项目的根目录。其中包含一个 core 目录是项目的 MVC 代码，其中的子目录结构如下。

- **ctls　存放相册系统中的控制器，包括前台的 front 和后台的 admin。**
- **include　存放系统中的一些公用函数和类。**
- **libs　存放系统中所用到的类库，像数据库访问类、输出类等。**
- **models　存放系统的模型层代码。**
- **views　存放系统的视图层（页面）代码，同样分为前台的 front 和后台的 admin 两类。**

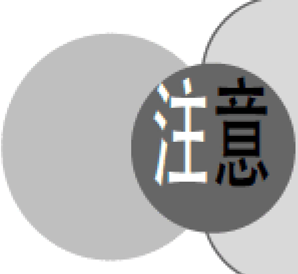
core 是整个相册系统的核心目录，后面介绍的很多代码都存放在此目录下。除此之外，在项目根目录下还有一个保存配置文件的 conf 目录以及保存上传图片的 data 目录。

12.2.2　设计通用类

在 MVC 模式中的三层之间是并列关系，因此任何一层都可以相互调用或者调用一些共用的函数和类。这些代码封装后存放在 core 的 include 目录中，像实现载入模型的类、前台页面的基类、重定向函数以及分页类等。下面介绍其中最重要的几个文件，其他文件可参考实例源代码。

（1）在 modelfactory.php 文件中定义一个 modelfactory 类。该类的代码虽然不多，仅包含一个 modelfactory()方法，但是却有很重要的作用，具体代码如下所示：

```
class modelfactory{
    function modelfactory(){
        $this->output =& get_output();        //获取要输出的内容
        $this->db =& db();                    //获取数据库
    }
}
```



在 PHP 5 以前的版本中，类的构造函数即是类的同名函数，而到了 PHP 5，类的构造函数多了一个__construct 函数。当这两个同时存在时，只调用__construct 函数。

（2）在 pgecore.php 文件中定义一个 pagecore 类。该类主要用于从 PHP 的全局数组中获取一个值，像$_POST、$_GET 和$_REQUEST，具体代码如下所示：

```
class pagecore{
    function isPost(){                              //判断是否为 post 方式提交
        if(strtolower($_SERVER['REQUEST_METHOD']) == 'post'){
            return true;
        }
        return false;
    }
    function getGet($key,$default=''){              //从$_GET 中获取值
        if(isset($_GET[$key])){
            if(!get_magic_quotes_gpc())
            {
                if(is_array($_GET[$key])){
                    return array_map('addslashes',$_GET[$key]);
                }else{
                    return addslashes($_GET[$key]);
                }
            }
            return $_GET[$key];
        }
        return $default;
    }
    function getPost($key,$default=''){ //从$_POST 中获取值
        if(isset($_POST[$key])){
            if(!get_magic_quotes_gpc())
            {
                if(is_array($_POST[$key])){
                    return array_map('addslashes',$_POST[$key]);
                }else{
                    return addslashes($_POST[$key]);
                }
            }
            return $_POST[$key];
        }
        return $default;
    }
    function getRequest($key,$default=''){  //从$_REQUEST 中获取值
        if(isset($_REQUEST[$key])){
            if(!get_magic_quotes_gpc())
```

```
            {
                if(is_array($_REQUEST[$key])){
                    return array_map('addslashes',$_REQUEST[$key]);
                }else{
                    return addslashes($_REQUEST[$key]);
                }
            }
            return $_REQUEST[$key];
        }
        return $default;
    }
}
```

（3）在 frontpage.php 文件中定义一个 frontpage 类。该类继承上面的 pagecore 类用于表示相册中的前台页面，具体代码如下所示：

```
require_once(INCDIR.'pagecore.php');          //引入 pagecore 类所在文件
class frontpage extends pagecore{             //继承 pagecore 类
    function frontpage(){
        global $setting;                      //获取全局设置
        $this->setting = $setting;
        $this->output =& get_output();        //获取输出
        $this->view = new View();             //获取视图
        $this->db =& db();                    //准备数据
        $this->output->set('site_title',$this->setting['site_title']);
                                              //指定页面变量的值
        $this->output->set('site_keyword',$this->setting['site_keyword']);
        $this->output->set('site_description',$this->setting
        ['site_description']);
    }
}
```

（4）在 adminpage.php 文件中定义一个 adminpage 类。该类也继承自 pagecore 类用于表示相册中的后台页面，具体代码如下所示：

```
require_once(INCDIR.'pagecore.php');          //引入 pagecore 类所在文件
class adminpage extends pagecore{             //继承 pagecore 类
    function adminpage(){
        global $setting;
        $this->setting = $setting;
        $this->output =& get_output();
        $this->view = new View();
        $this->db =& db();
        $this->auth = new auth();              //判断是否有权限
        $this->output->set('open_photo_setting',$this->setting
        ['open_photo']);
```

```
    }
}
```

12.2.3 设计类库

与通用类不同，在类库中封装了对某一对象的所有操作，例如前面用到的管理员权限验证类 auth、视图类 view 以及数据库操作类等。这些类库都存放在 core 的 libs 目录中，下面介绍其中最常用的类。

（1）在 output.class.php 文件中定义一个 output 类。该类用于在输出时从数组中获取一个值或者添加一个值，具体代码如下所示：

```
class output{
    var $data = array();

    function set($key,$value){
        $this -> data[$key] = $value;
    }

    function get($key = ''){
        if(!$key) return $this -> data;
        else  return isset($this->data[$key]) ? $this->data[$key] : '';
    }
}
```

（2）在 view.class.php 文件中定义一个 View 类。该类表示一个视图（即一个页面），该视图通常都是预定义好的模板文件，具体代码如下所示：

```
class View{
    var $tplFile;                          //指定模板文件
    function View($tplFile = '')    {
        if(!empty($tplFile))
            $this -> tplFile =  "{$tplFile}";
    }
    /**
     * 返回待输出的 view 内容
     */
    function fetch($tplFile = ''){
        if(!empty($tplFile))
            $this -> tplFile =  "{$tplFile}";

        @ob_clean();
        ob_start();
        //模板中直接使用的对象
        $res =& get_output();
        if(file_exists(VIEWDIR.$this->tplFile))
```

据编号修改管理员的密码。其他模型类的代码这里就不再重复，请读者参考实例源代码。

12.2.5 配置文件

上面介绍的都是 core 目录下有关 MVC 的具体实现，本节介绍的是系统的配置文件，它们存放在站点的 conf 目录下。

在本系统中主要包括两个配置文件，第一个 config.php 文件保存的是系统的数据库配置信息，像使用的连接类库、MySQL 主机名、登录账号、系统数据库和字符集编码等。如下所示是 config.php 文件的具体定义：

```
$db config = array(
'adapter'  => 'mysqli',
'host'     => 'localhost',
'port'     => '3306',
'dbuser'   => 'root',
'dbpass'   => '123456',
'dbname'   => 'db_gallery',
'pconnect' => false,
'charset'  => 'utf8',
'pre'      => 'my '
);
```

第二个 setting.php 文件保存的是系统中有关相册的全局配置，像相册名称、关键字、访问的 URL、每页显示数量、是否使用水印等。如下所示是该文件的具体代码：

```
$setting['site title'] = '我的相册';
$setting['site_keyword'] = '';
$setting['site_description'] = '';
$setting['url'] = 'http://localhost/mygallery/';
$setting['open pre resize'] = false;
$setting['resize img width'] = '1600';
$setting['resize img height'] = '1200';
$setting['resize quality'] = '100';
$setting['demand_resize'] = false;
$setting['imgdir_type'] = '2';
$setting['size allow'] = '1024000';
$setting['pageset'] = '15';
$setting['open photo'] = true;
$setting['gallery limit'] = '60';
$setting['access ctl'] = false;
$setting['access domain'] = 'localhost';
$setting['open_watermark'] = false;
$setting['watermark_path'] = '';
$setting['watermark_pos'] = 0;
```

12.3 前台功能实现

经过前面的设计和操作，系统的基本运行环境就搭建完成了。接下来的工作就是在这个基础上实现相册的功能。这里首先从相册的前台开始，它的使用频率最高，主要包括提供一个相册列表、单击可查看相册下的所有照片，以及单击照片查看原图等。

12.3.1 查看所有相册

在首页默认会显示系统中所有的相册，并显示相册的封面图片、创建时间和相册名称等，运行效果如图 12-2 所示。

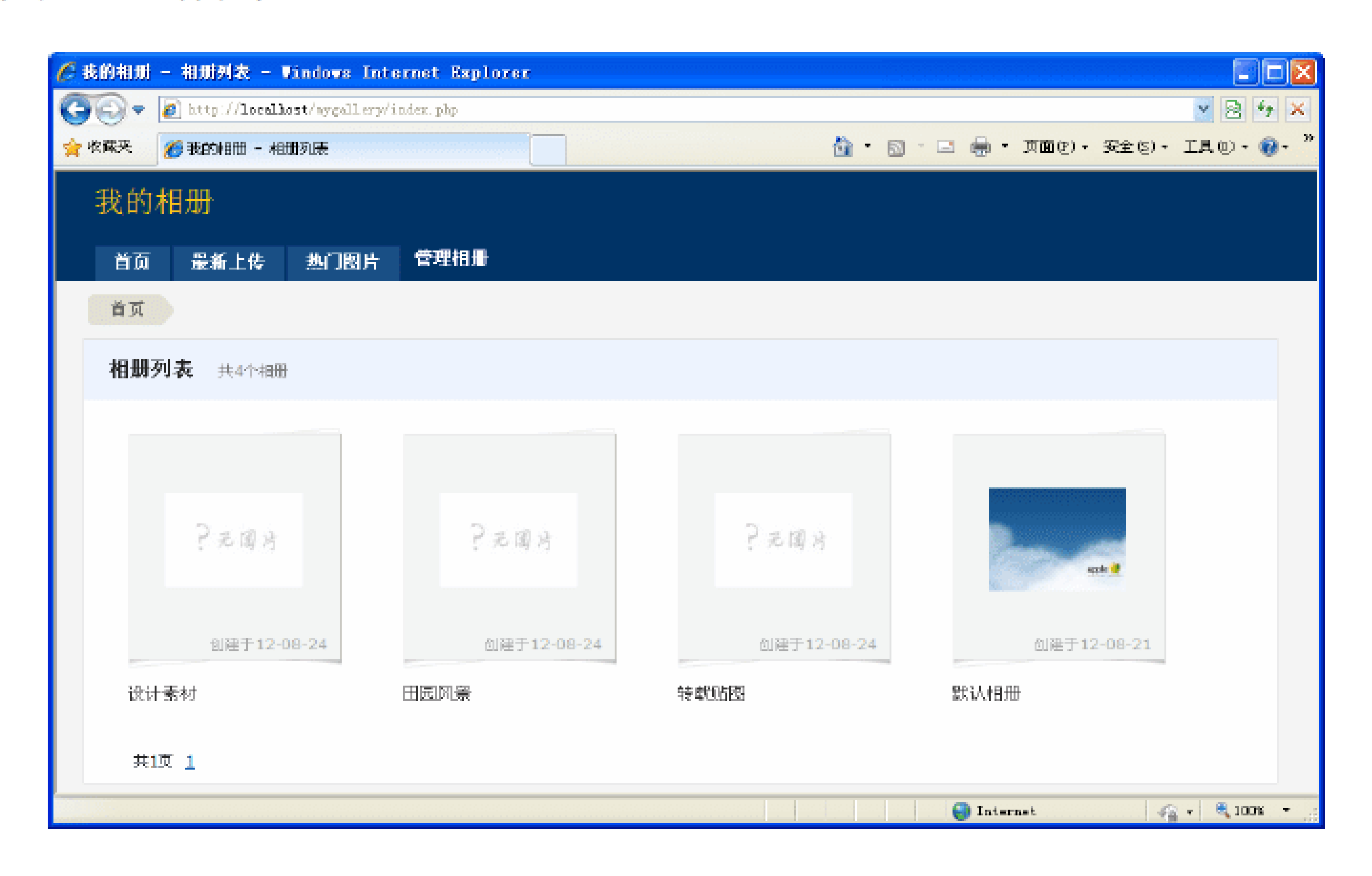


图 12-2　首页效果

从图 12-2 可以看出，首页是由 index.php 文件实现的，它会去调用前台首页控制器的 index 动作。前面介绍过项目的控制器位于 core 的 ctls 目录中，该目录下又分为 front（前台）和 admin（后台）两个子目录。

首页控制器的具体实现文件是 core\ctls\front\default.php。该文件其实定义了一个继承自 frontpage 类的 controller 子类，在 controller 类中又包含了一个 index 函数表示默认的首页请求动作，这部分代码如下所示：

```
//core\ctls\front\default.php 文件
class controller extends frontpage{                    //表示前台首页的控制器
    function index(){                                  //默认请求时的动作
        $this->mdl_album = & load_model('album');      //载入相册模型
        $page = $this->getGet('page',0);               //是否有分页参数
        if(!$page){
            $page = 1;                                 //如果没有默认为第一页
        }
        //调用模型的方法获取数据
        $albums = $this->mdl_album->get_all_album($page,true);
```

```
        $pageurl='index.php?ctl=album&page=[#page#]';
        //准备数据
        if($albums['ls']){
            foreach($albums['ls'] as $k=>$v){
                $cover = $this->mdl album->get cover($v['id'],$v['cover']);
                $albums['ls'][$k]['cover']=$cover?mkImgLink($cover['dir'],
                $cover['pickey'],$cover['ext'],'thumb'): 'img/nopic.jpg';
            }
        }
        //设置视图中可用的变量
        $this->output->set('current nav','index');
        $this->output->set('albums',$albums['ls']);
        $this->output->set('pageset',pageshow($albums['total'],
        $albums['start'],$pageurl));
        $this->output->set('total_num',$albums['count']);

        $site_title = $this->output->get('site_title').' - 相册列表';
        $this->output->set('site title',$site title);
        //显示指定的视图文件
        $this->view->display('front/album.php');
    }
}
```

index()函数执行流程是首先初始一个表示相册的模型，然后调用模型提供的方法获取指定页（默认为第 1 页）的数据并保存到$ablums 变量中。然后对该变量进行遍历准备需要显示的数据，然后设置一些需要显示的其他数据，最后调用 display()方法将数据关联到指定的视图（模板）文件。

在 index()函数用到的相册模型 album 存放在 core\ctls\models\albumn.php 文件中。album 类继承自 modelfactory 类，上面调用的是 get_all_album()方法，具体实现代码如下：

```
//core\ctls\models\album.php 文件
class album extends modelfactory{        //album 类继承自 modelfactory 类
    //获取所有相册的方法
    //$page 参数要获取第几页的数据，默认为空表示所有
    //$fileter_private 参数是否隐藏私有相册，默认为 false
    //返回值按相册编号降序排列返回数据数组
    function get all album($page = NULL,$filter private=false){
        if($filter_private){                //是否过滤私有相册
            $where = 'private=0';
        }else{
            $where = '';
        }
        $this->db->select('#albums',"*",$where,'id desc');    //指定查询
        if($page){                                          //获取全部还是指定页的数据
            $pics = $this->db->toPage($page,PAGE_SET);
```

```
        }else{
            $pics = $this->db->getAll();
        }
        return $pics;                                        //返回结果
    }
}
```


如上述代码所示，get_all_album()方法的实现代码比较简单，这里就不再详细解释。下面来看一下图 12-2 所示的视图，它位于 core\views\front\album.php 文件。

在相册的视图中主要有 3 个部分需要显示，第一个就是当前相册的数量，这需要用到控制器中的 total_num 变量，实现代码如下所示：

```
<div class="col titbg">
  <h1 class="f left">相册列表</h1>
  <h4 class="f_left">共<?php echo $res->get('total_num');?>个相册</h4>
</div>
```

第二部分是遍历控制器中提供的相册数组，然后输出到页面，这部分代码如下所示：

```
    <div class="gallary_wrap">
      <?php $ls = $res->get('albums');                     //获取数组
    if($ls):                                                //如果数组不为空
    foreach($ls as $k=>$v):                                 //进行遍历
    ?>
      <div class="gallary item album">
        <div class="item">
          <div class="pic_box">
            <table>
              <tr>
                <td><a href="index.php?ctl=album&act=photos&album=<?
php echo$v['id']; ?>"><img src="<?php echo $v['cover'];?>" /></a></td>
               </tr>
             </table>
           </div>
           <div class="clear brief"> <span class="f_left"></span> <span
          class="f_right gray"> 创建于<?
php echo date("y-m-d",$v['create time']);?></span> </div>
          </div>
          <div class="info">
            <div class="title"> <span class="name">
            <a href="index.php?ctl=album&act=photos&album=<?php
            echo $v['id']; ?>"> <?php echo $v['name'];?></a> </span> </div>
          </div>
        </div>
        <?php
      endforeach;                                         //结束遍历
```

```
        else:
        ?>
          <div class="gallary item album">
            <div class="item">
              <div class="pic box">
                <table>
                  <tr>
                    <td>当前没有任何相册！ </td>
                  </tr>
                </table>
              </div>
            </div>
          </div>
          <?php
        endif;
        ?>
        </div>
```

第三部分是在相册列表下方的分页导航，它的实现代码非常简单，如下所示：

```
    <div class="f_left paginator"> <?php echo $res->get('pageset');?> </div>
```

12.3.2　查看相册图片

在首页中单击一个相册即可查看该相册下的所有图片，例如查看默认相册的运行效果如图 12-3 所示。

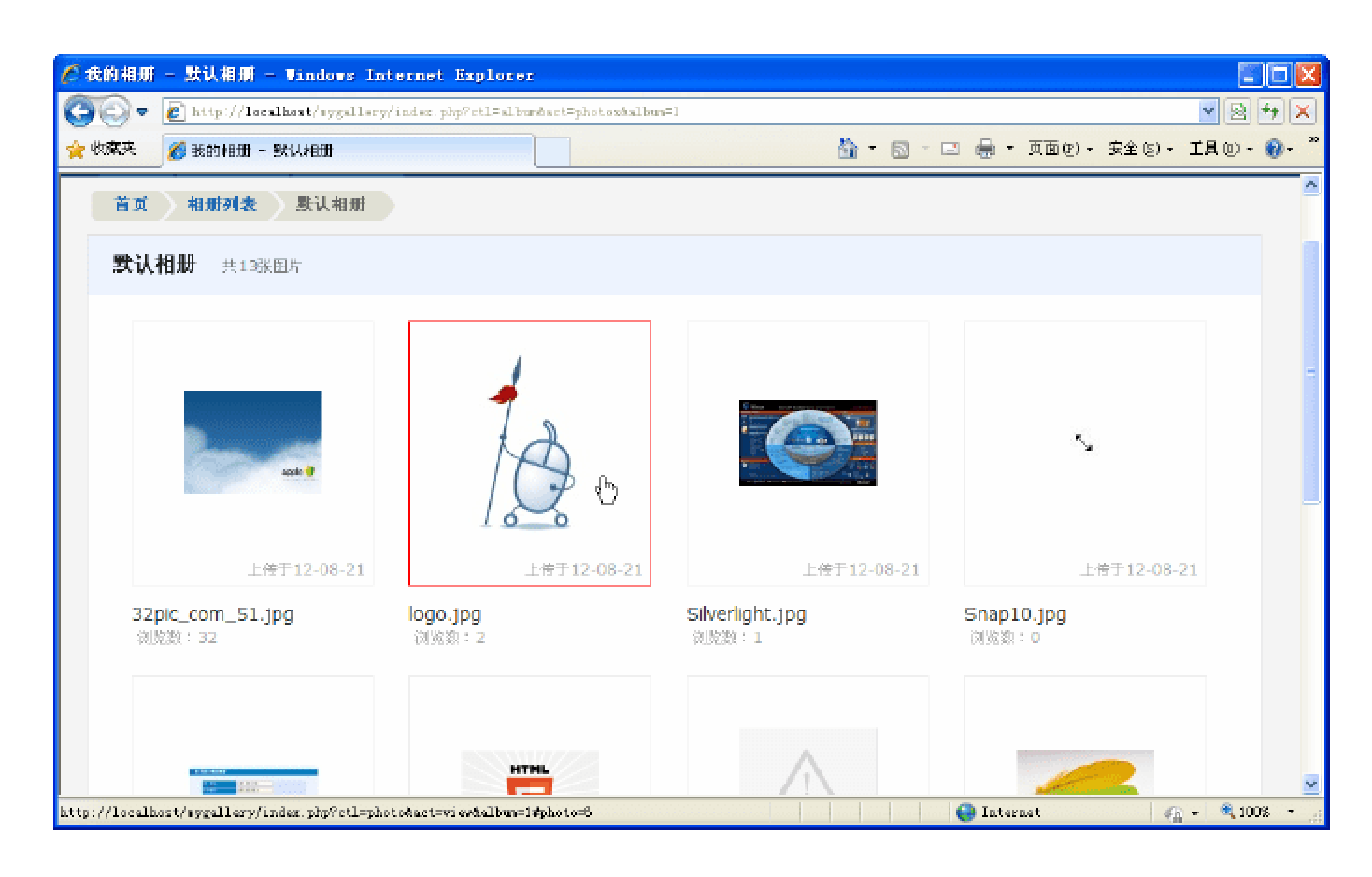


图 12-3　查看相册下所有图片运行效果

这个页面与相册页面使用的是相同的控制器，位于 core\ctls\front\album.php 文件中。不同的是这里查看相册下的所有图片调用的是 photos()方法，它的代码如下所示：

```
    // core\ctls\front\album.php 文件
       function photos(){                              //获取相册下的所有图片
```

```
        $album = intval($this->getGet('album',0));   //获取相册编号
        $page = $this->getGet('page',0);         //获取显示页数
        if(!$page){
            $page = 1;
        }
        //调用 picture 模型的 get all pic()方法获取数据
        $pics = load_model('picture')->get_all_pic($page,$album,
        'time_asc');
        //定义页面的 URL
        $pageurl="index.php?ctl=album&act=photos&album={$album}
        &page=[#page#]";
        //准备数据
        $album name = $this->mdl album->get album name($album);
        $this->output->set('current nav','index');
        $this->output->set('piclist',$pics['ls']);
        $this->output->set('album_name',$album_name);
        $this->output->set('album',$album);
        $this->output->set('pageset',pageshow($pics['total'],
        $pics['start'],$pageurl));
        $this->output->set('total num',$pics['count']);

        $site_title = $this->output->get('site_title').' - '.$album_name;
        $this->output->set('site_title',$site_title);
        //显示视图
        $this->view->display('front/album photos.php');
    }
```

photos()方法实现思路与 index()方法相同，都需要先载入模型，然后调用模型的方法获取数据，再对数据进行组织和关联变量，最后输出到视图显示。

photos()方法用到的照片模型 picture 位于 core\models\picture.php 文件中，具体代码就不再给出，可以参考实例源码。下面重点看看对应的视图文件 core\views\ front\album_photos.php。

首先在视图中需要显示当前正在浏览的相册名称，实现代码如下所示：

```
<ul class="png">
    <li class="png"><a href="index.php">首页</a></li>
    <li class="png"><a href="index.php?ctl=album">相册列表</a></li>
    <li class="png last"><span><?php
     echo $res->get('album name');?></span></li>
</ul>
```

在上述代码中调用控制器中的 album_name 变量获取相册名称。该名称还将显示在下方的标题中，同时显示的还有相册中照片的数量，实现代码如下所示：

```
<div class="col titbg">
   <h1 class="f_left"><?php echo $res->get('album_name');?></h1>
```

```
    <h4 class="f_left">共<?php echo $res->get('total_num');?>张图片</h4>
  </div>
```

接下来编写代码获取所有照片数据并进行遍历和输出，实现代码如下所示：

```
      <div class="gallary wrap">
        <?php $ls = $res->get('piclist');              //获取所有照片数据
  if($ls):
  foreach($ls as $k=>$v):                              //遍历照片数组
  ?>
        <DIV class=gallary_item>
          <DIV class=item>
            <DIV class=pic box>
              <TABLE>
                <TBODY>
                  <TR>
                    <TD><a   href="index.php?ctl=photo&act=view&album=<?php
echo  $v['album'];  ?>#photo=<?php  echo  $v['id'];?>"><img  src="<?php  echo
mkImgLink($v['dir'],$v['pickey'],$v['ext'],'thumb');?>" /></a></TD>
                  </TR>
                </TBODY>
              </TABLE>
            </DIV>
            <DIV class=pic ctl>
              <UL class=btns>
              </UL>
            </DIV>
            <DIV   class="clear   brief"><SPAN   class=f left></SPAN><SPAN
class="f_right  gray"> 上 传 于 <?php  echo  date("y-m-d",$v['create_time']);?>
</SPAN></DIV>
           </DIV>
           <DIV class=info>
           <DIV class=title><SPAN class=name><a href="index.php? ctl=photo-
&act=view&album=<?php echo $v['album']; ?> #photo=<?php echo $v['id'];?>"><?php
echo $v['name'];?></a></SPAN> </DIV>
           <DIV class=info_col><SPAN class="f_left gray">浏览数：<?php echo
$v['hits'];?></SPAN><SPAN class="f right gray"></SPAN></DIV>
          </DIV>
        </DIV>
        <?php
      endforeach;
      else:
      ?>
        <div class="gallary item album">
          <div class="item">
            <div class="pic_box">
```

```
            <table>
              <tr>
                <td>当前没有任何图片！ </td>
              </tr>
            </table>
          </div>
        </div>
      </div>
      <?php
    endif;
    ?>
    </div>
```

最后是输出一个分页列表，代码与 12.3.1 节相同就不再重复。

12.3.3 查看图片详情

相册系统的最主要功能就是查看图片，前面的查看相册列表和查看图片列表都是为查看图片作准备。当浏览相册图片时进行单击，即可查看该图片的原始大小以及更多的详细信息，如图 12-4 所示。

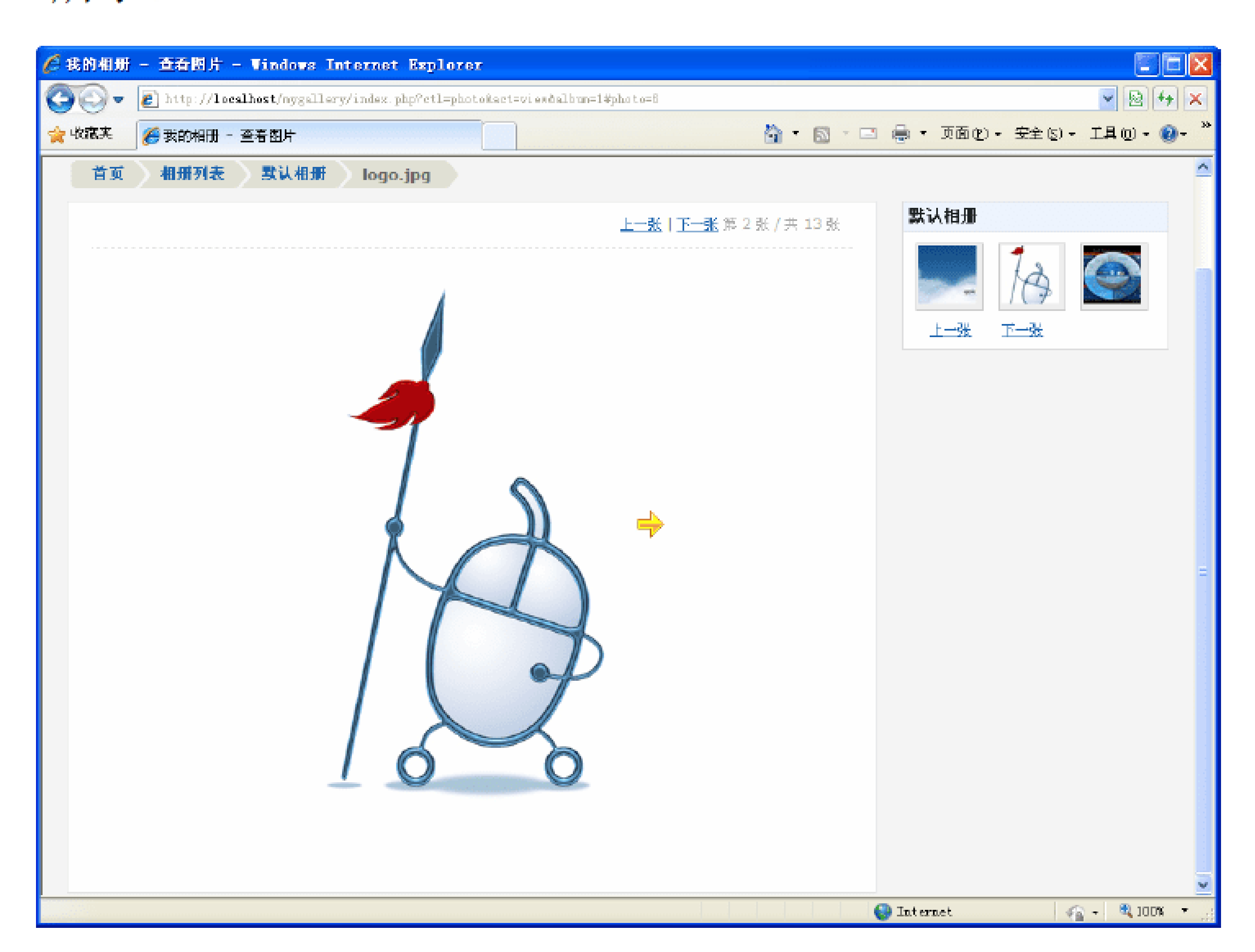


图 12-4 查看图片详细信息

在这里不仅可以查看图片详情，还可以单击向左或者向右箭头快速浏览上一张或者下一张图片。

这里调用的控制器位于 core\ctls\front\photo.php 文件，请求的是 view()方法，具体代码如下所示：

```
//core\ctls\front\photo.php 文件
class controller extends frontpage{
```

```
    //查看图片详情的方法
    function view(){
        $album = intval($this->getGet('album'));            //获取相册编号
        //获取属于该相册的所有照片
        $picls =$this->mdl picture->get all pic(null,$album,
       'time asc','0',true);
        $mini_photo_width = (count($picls)+5)*65;
        //准备数据
        $this->output->set('total imgs',count($picls));
        $this->output->set('piclist',$picls);
        $this->output->set('mini photo width',$mini photo width);
        $this->output->set('album name',$this->mdl album->get album name
        ($album));
        $this->output->set('album',$album);

        $site_title = $this->output->get('site_title').' - 查看图片';
        $this->output->set('site title',$site title);
         //关联视图
        $this->view->display('front/viewphoto.php');
    }
}
```

这里除了使用 album 模型外还使用了 picture 模型。在模型里封装了获取数据的方法，控制器则调用这些方法将数据与视图进行关联。下面来看看这里用到的视图文件 core\views\front\viewphoto.php。

首先需要显示的是当前图片在相册中的位置、相册中图片的总数量、上/下一张链接，这部分代码如下所示：

```
<div id="pic_block" class="tbmu">
   <div  class="f right">  <a  href="javascript:void(0)"  class="btnprev"
onclick="showPre()"> 上 一 张 </a>  <span  class="pipe">|</span>  <a
href="javascript:void(0)" onclick="showNext()" class="btnnext">下一张</a>
   <span class="gray" id="show_photo_page"> 第 <span class="cur">0</span> 张
/ 共 <?php echo $res->get('total imgs'); ?> 张 </span> </div>
 </div>
```

对于图片的显示，这里使用的是 JavaScript 脚本，因此只需要进行定义布局即可，从而减轻服务器的压力。代码如下所示：

```
<div id="photo_body">  <span id="imgarea" class="photo"></span>  </div>
```

最终的图片将以 HTML 的 img 标记形式插入到 ID 是 imgarea 的 span 标记内。实例中使用的是 jQuery 框架，由于代码比较多就不再给出，可以参考实例源代码。

下面来看看图 12-4 中右侧图片滚动列表的具体布局，代码如下所示：

```
 <div class="pic_nav">
```

```
        <h2 class="titbg"><?php echo $res->get('album_name'); ?></h2>
        <div class="pic_nav_body" id="miniphoto_list">
          <ul class="thumblist">
             <?php $ls = $res->get('piclist');
                if($ls):
                    foreach($ls as $k=>$v):
                ?>
                <li id="li_<?php echo $k;?>" rel="<?php echo $v['id'];?>">
                    <a href="javascript:void(0)" onclick="return showPhoto(<?php
echo $k;?>);" onfocus="this.blur()"><img id="i_<?php echo $k;?>" width="50"
height="50"  title="<?php  echo  $v['name'];  ?>"  rel="<?php  echo
mkImgLink($v['dir'],$v['pickey'],$v['ext'],'big');?>"    src="about:blank"
csrc="<?php  echo  mkImgLink($v['dir'],$v['pickey'],$v['ext'],'square');?>"
/></a>
                </li>
                <?php
                    endforeach;
                endif;
                ?>
          </ul>
          <div class="nav_ctl"> <span class="nav_ctl_prev"><a href="javascript:
     void(0)" onclick="showPre()">上一张</a></span> <span class="nav_ctl_next">
     <a href="javascript:void(0)" onclick="showNext()" >下一张</a></span>
             <div class="clear"></div>
           </div>
         </div>
       </div>
```

这里同样使用了 jQuery 来动态响应用户的单击切换图片，甚至按键盘的向左或者向右键也可以切换。

12.3.4 随便看看

随便看看就是指不通过相册快速查看系统中的图片，这里提供了两种方式分别是查看最新上传的图片和查看最热门的图片。

首先来看看最新上传图片的运行效果，如图 12-5 所示。

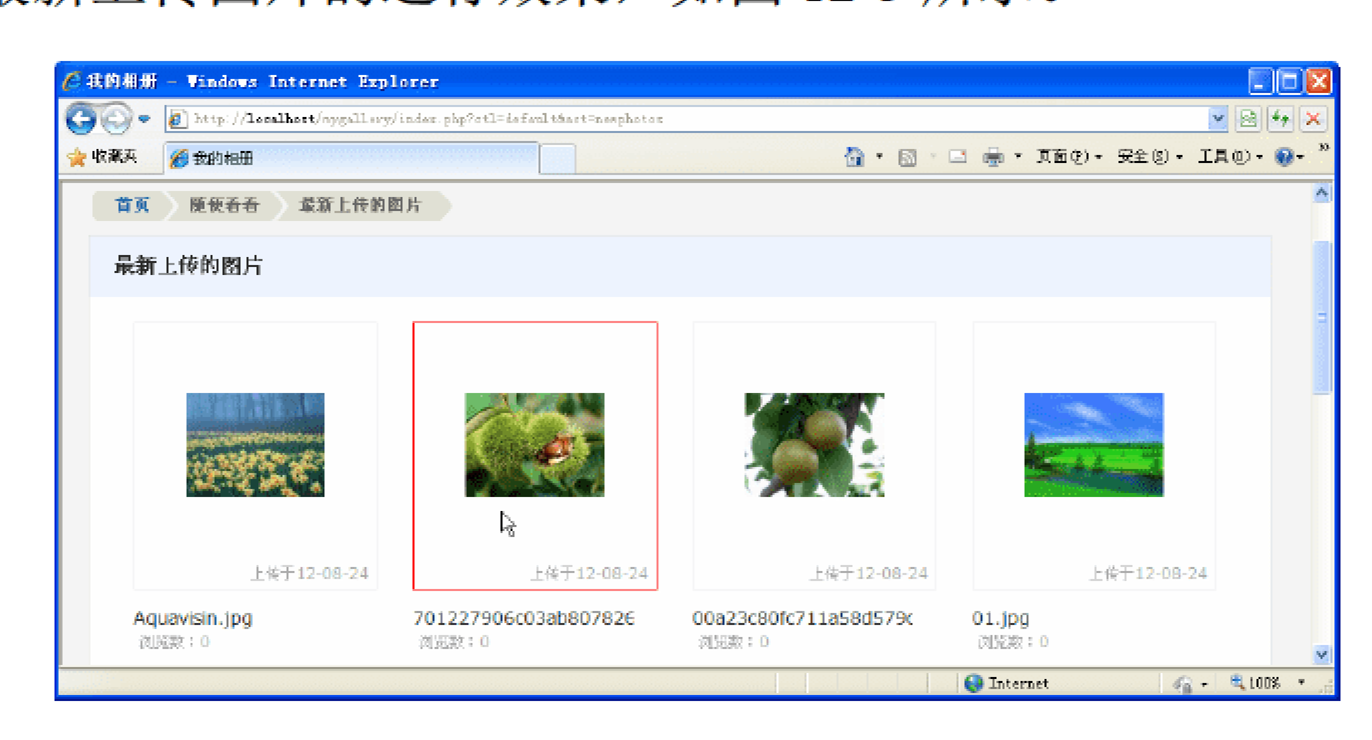


图 12-5　查看最新上传图片

它与相册列表调用的是相同的控制器，不同的是这里请求的是 newphotos()方法，该方法代码如下所示：

```
//core\ctls\front\defaut.php 文件
    function newphotos(){                              //最新上传图片
        $page = $this->getGet('page',0);
        if(!$page){
            $page = 1;
        }
        $pageurl='index.php?act=newphotos&page=[#page#]';
        $mdl picture = & load model('picture');
        $piclist = $mdl_picture->get_all_pic($page,0,'time_desc',0,true);
        $this->output->set('piclist',$piclist['ls']);
        $this->output->set('pageset',pageshow($piclist['total'],
        $piclist['start'],$pageurl));
        $this->output->set('total num',$piclist['count']);
        $this->output->set('current nav','newphotos');
        $this->view->display('front/newphotos.php');
    }
```

可以看到这里使用的是 picture 模型，视图文件是 core\views\front\newphoto.php，具体的布局与相册列表相同，也就不再重复。

接下来看看查看热门图片的运行效果，如图 12-6 所示。

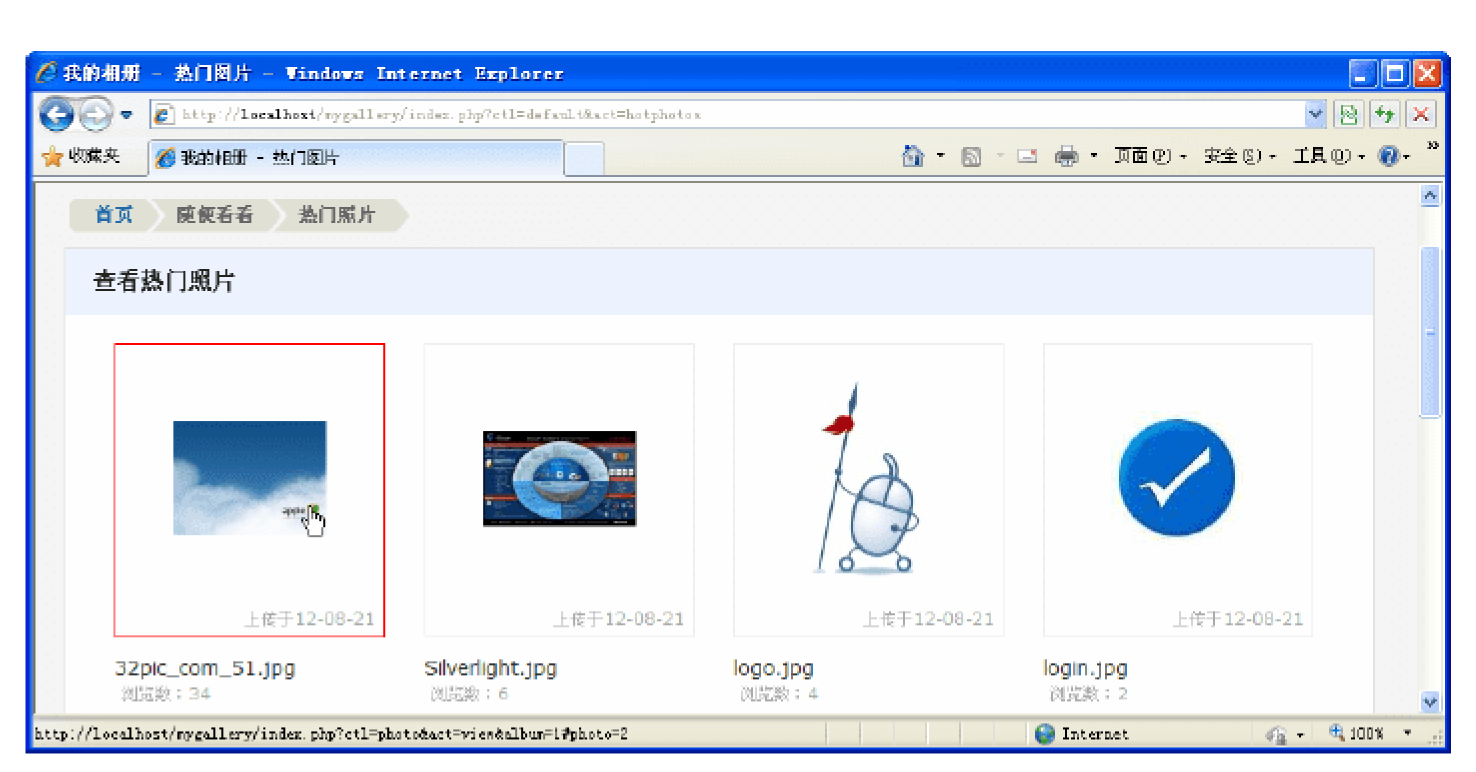


图 12-6　查看最新热门图片

它也是使用相册列表的控制器，请求的方法是 hotphotos()方法，实现代码如下所示：

```
//core\ctls\front\defaut.php 文件
    function hotphotos(){                              //最热门图片
        $this->output->set('current nav','hotphotos');

        $mdl picture = & load model('picture');
        $piclist = $mdl_picture->get_all_pic(NULL,0,'hot',10,true);
        $this->output->set('piclist',$piclist);
        $site_title = $this->output->get('site_title').' - 热门图片';
        $this->output->set('site_title',$site_title);
```

```
        $this->view->display('front/hotphotos.php');
    }
```

从代码中可以看到，热门图片就是按照浏览次数降序排列，并显示前 10 张图片。布局代码与前面相同就不再重复。

12.4 管理员登录

至此，相册系统的前台功能就算完成了。为了区分相册前台的普通用户与后台的管理员身份，在系统中设置了登录验证，只有验证成功后才能执行管理操作。

在网站根目录下新建 admin.php 文件表示后台的登录入口文件。然后在 admin.php 文件中调用后台控制器的 login()方法表示登录请求。

后台控制器位于 core\ctls\admin\default.php 文件中，其中 login()方法的代码如下：

```
//core\ctls\admin\defaut.php 文件
class controller extends adminpage{              //后台控制器类
    function login(){                            //登录调用的方法
        if($this->isPost()){                     //是否提交表单
            $username = addslashes($this->getPost('loginname'));
            $userpass = $this->getPost('operatorpw');
            $remember = $this->getPost('remember');
            if($remember){
                $expire_time = time()+86400*365;
            }else{
                $expire_time = 0;
            }
            if($this->auth->setLogin($username,md5($userpass),
            $expire_time)){
                redirect_c('album','index');     //验证成功则转向管理页面
            }else{
                redirect('admin.php?ctl=default&act=login&flag=1');
            }
        }else{                                   //显示登录表单
            $flag = $this->getGet('flag');
            $this->output->set('flag',$flag);
            $this->view->display('admin/login.php');
        }
    }
}
```

如上述代码所示，后台控制器 controller 类继承自 adminpage 类。第一次调用 login()方法时，由于没有提交此时显示登录表单，视图文件是 core\viewes\admin\login.php。

登录表单的主要代码如下：

```
  <form method="post"  action="admin.php?ctl=default&act=login"
```

```
 name="login_soft">
      <table width="200" border="0" cellspacing="0" cellpadding="0">
        <tr>
          <td height="40" colspan="2"><input type="text" name="loginname"
id="user" ></td>
        </tr>
        <tr>
          <td height="50" colspan="2"><input type="password" id="pwd"
name="operatorpw" >
            <br/><input type="checkbox" name="remember" value="1" />  记住密码
</td>
        </tr>
        <tr>
          <td height="60" align="left"><input name="loginBtn" type="submit"
id="loginBtn" value=" " /></td>
          <td height="60" align="right"><input name="resetBtn" type="reset"
id="resetBtn" value=" " /></td>
        </tr>
      </table> <br />
      <span style="font-weight:bold;color:#F00">
      <?php switch($res->get('flag')){
            case '1':
                echo '用户名或者密码错误！';
                break;
            case '2':
                echo '您已成功退出！';
                break;
            default:
                echo '欢迎登录相册管理系统！';
        }?>
      </span>
    </form>
```

代码提供了两个输入框、一个记住密码选项以及两个提交按钮，运行效果如图 12-7 所示。

单击“登录”按钮后表单仍然提交到上面的 login()方法，如果成功会转到管理首页，否则显示错误信息，如图 12-8 所示。

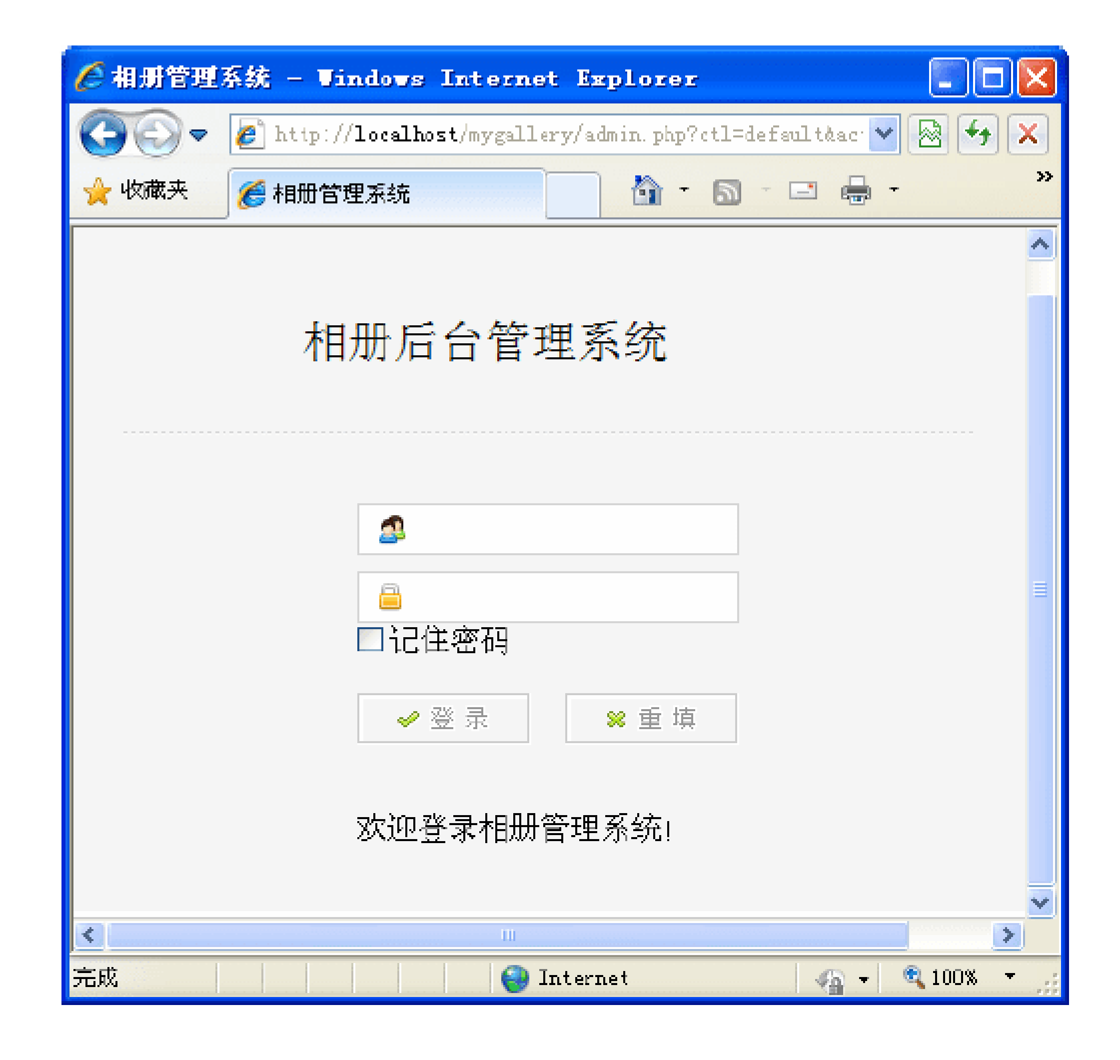


图 12-7 登录表单



图 12-8 登录验证失败

12.5　后台功能实现

在登录页面验证成功之后会将请求转到后台管理首页，它是由 core\ctls\admin\album.php 文件的控制器实现，请求的方法是 index()，对应的视图文件是 core\views\admin\album.php。

下面将对相册的后台管理功能进行介绍，主要包括创建一个相册、上传图片、删除图片、移动图片到其他相册，以及相册的重命名等。

12.5.1　创建相册

从系统前台的实现过程可以得知，相册在本系统的作用非常重要。它不仅是系统的顶层元素，而且是最终照片的容器。因此首先需要创建相册，方法是在后台管理首页中单击“创建相册”按钮，然后在弹出的浮动层中设置相册名称和是否私有，最后单击“确定”按钮完成创建，如图 12-9 所示。

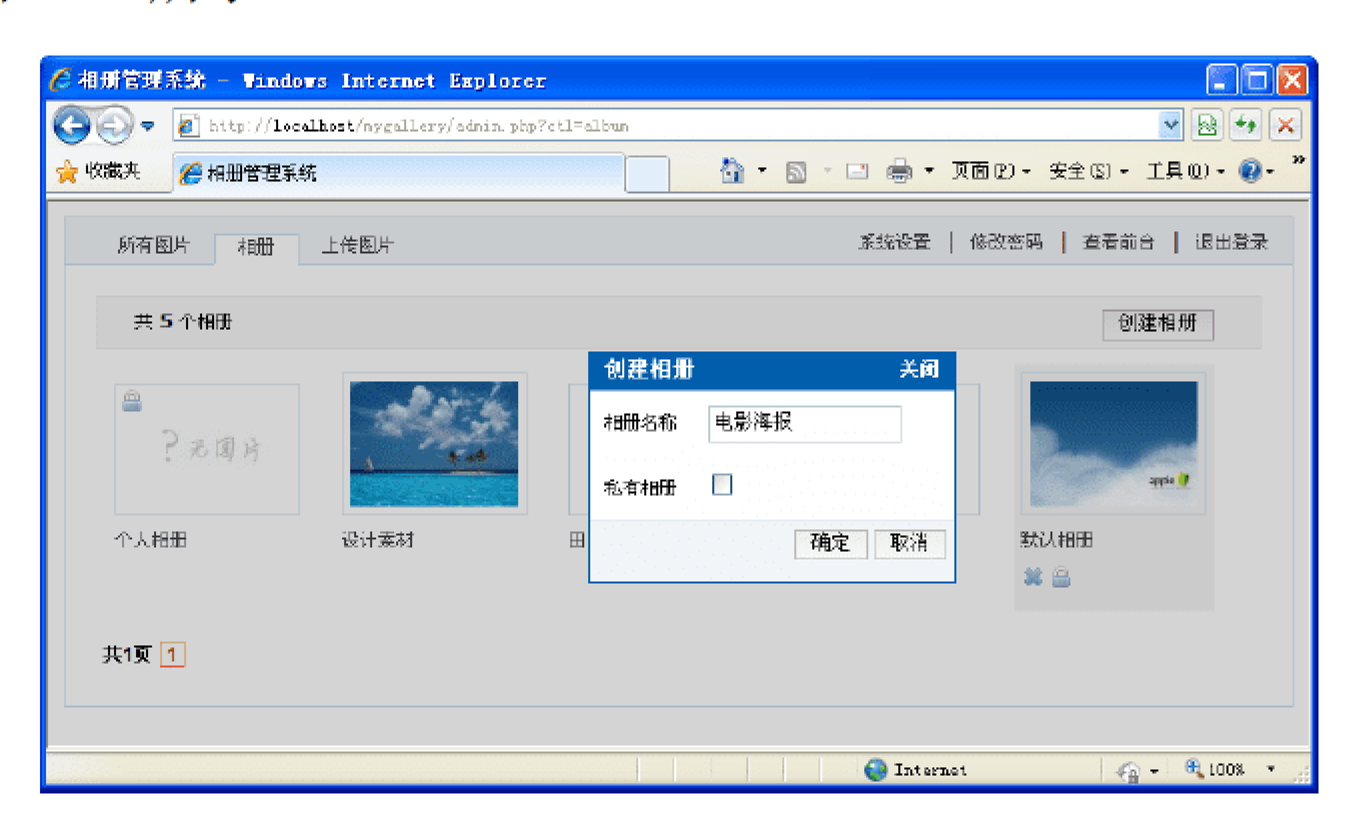


图 12-9　创建相册

这里请求的控制器位于 core\ctls\admin\album.php 文件，请求的方法为 index()，显示视图位于 core\views\admin\album.php。如下所示为视图中“创建相册”按钮的代码：

```
    <input type="button" onclick="create_album(1)" value="创建相册" class="btn">
```

“创建相册”按钮调用了 jQuery 函数 create_album()生成创建相册的布局，如下所示是该函数代码：

```
function create album(t){
    if(t==1){
        var func = 'do create album a()';
    }else{
        var func = 'do create album()';
    }
```

```
    function ajax_create_album(){                          //实现创建相册的方法
        $album name = $this->getPost('album name','');
        $album private = intval($this->getPost('album private',0));
        @ob clean();
        if($this->mdl album->insert album(array('name'=>$album name,
        'private'=>$album private,'create time'=>time() ))){
            $list = $this->mdl_album->get_albums_assoc();

            echo json encode(array('ret'=>true,'list'=>$list));
         }else{
             echo json_encode(array('ret'=>false,'msg'=>'创建相册失败！'));
         }
    }
```

12.5.2　上传图片

在相册前台中查看图片是最重要的功能之一，而上传图片则是相册后台中最重要的功能之一。通过上传，图片管理员可以向相册中添加新图片，从而更新相册。

上传图片使用的控制器位于 core\ctls\admin\upload.php 文件，根据请求方法的不同显示的视图文件也不一样。默认情况下会请求 index() 方法，它将显示 core\views\admin\upload_step1.php 视图文件让用户选择一个相册，运行效果如图 12-10 所示。

在如图 12-10 所示的界面下同样可以创建新相册。选择一个已经存在的相册之后单击“下一步”按钮向相册中选择要上传的图片，运行效果如图 12-11 所示。这是调用的 step2() 方法和 core\views\admin\upload_step2.php 视图文件。

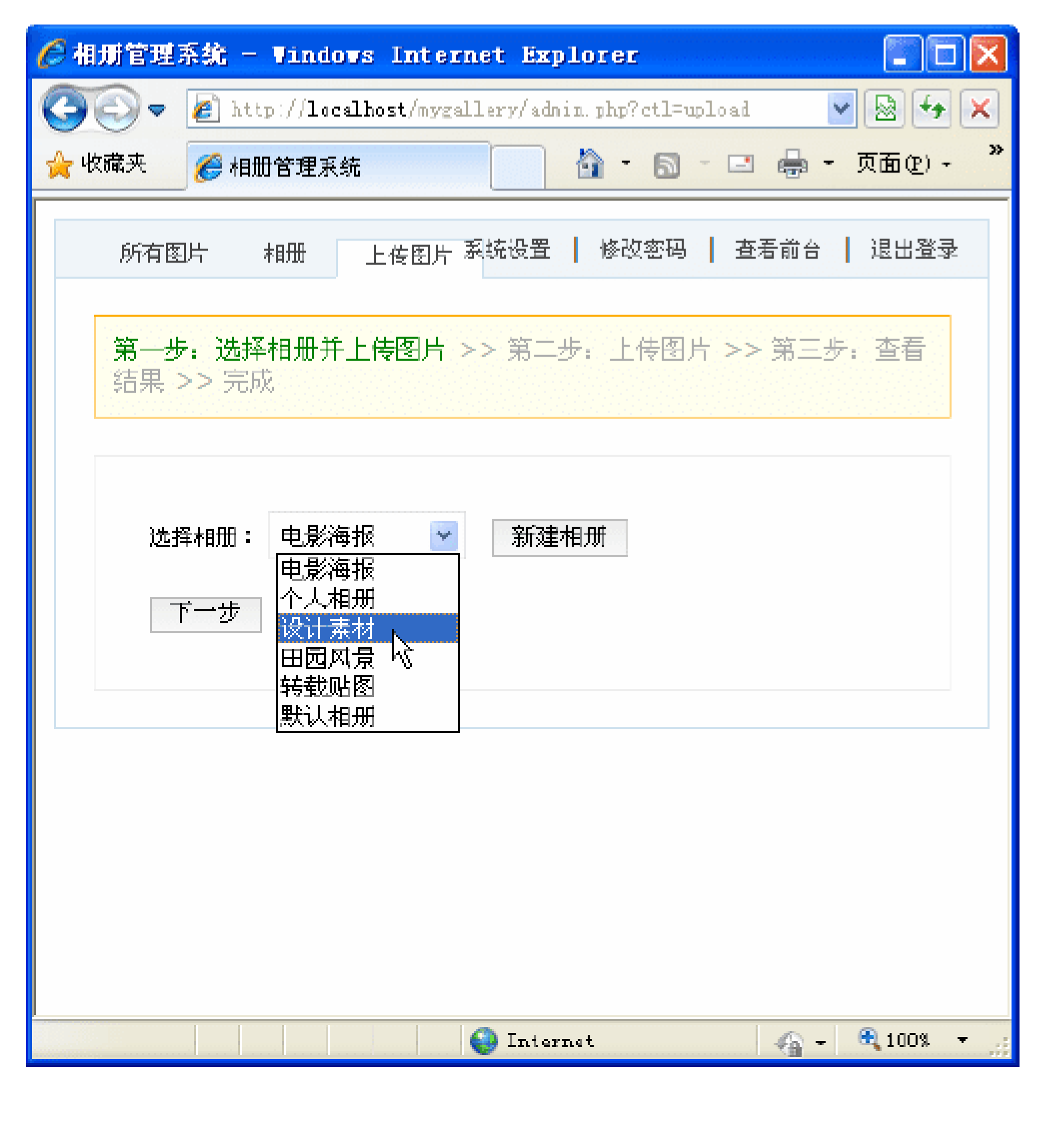


图 12-10　选择相册



图 12-11　选择要上传的图片

选择图片之后单击“开始上传”按钮，开始上传图片的过程，此时会显示每个文件的

这里请求的 photo 控制器位于 core\ctls\admin\photo.php 文件，请求的方法是 view()，用于显示的视图文件是 core\views\admin\viewphoto.php。

选择一个相册之后可以对其中的照片进行管理操作，它们都使用 core\ctls\admin\album.php 文件中的控制器，只是请求的方法不同。图 12-15 所示为选中几张图片进行批量删除的效果。

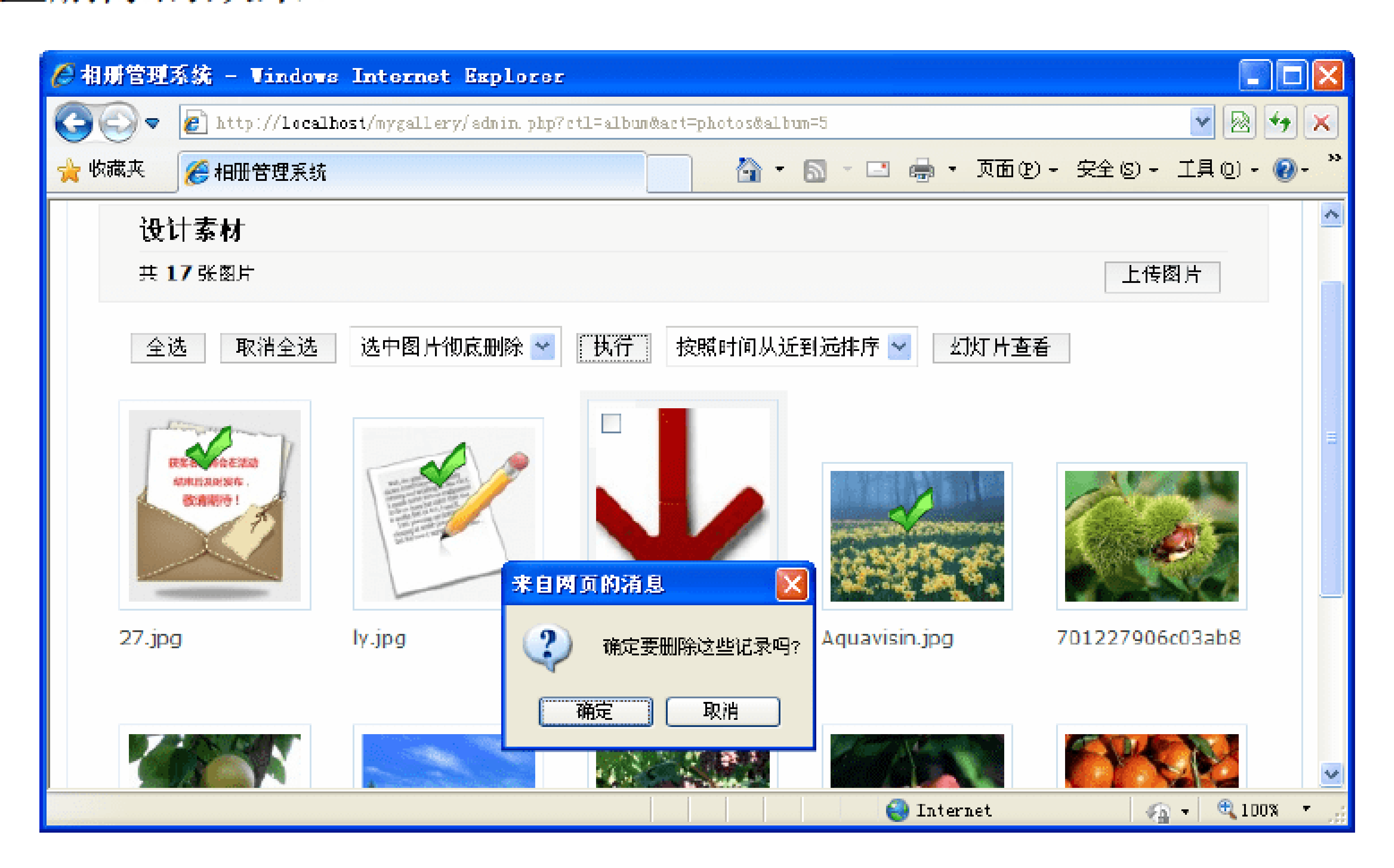


图 12-15　批量删除图片

也可以批量将选中的图片移动到其他相册，如图 12-16 所示。

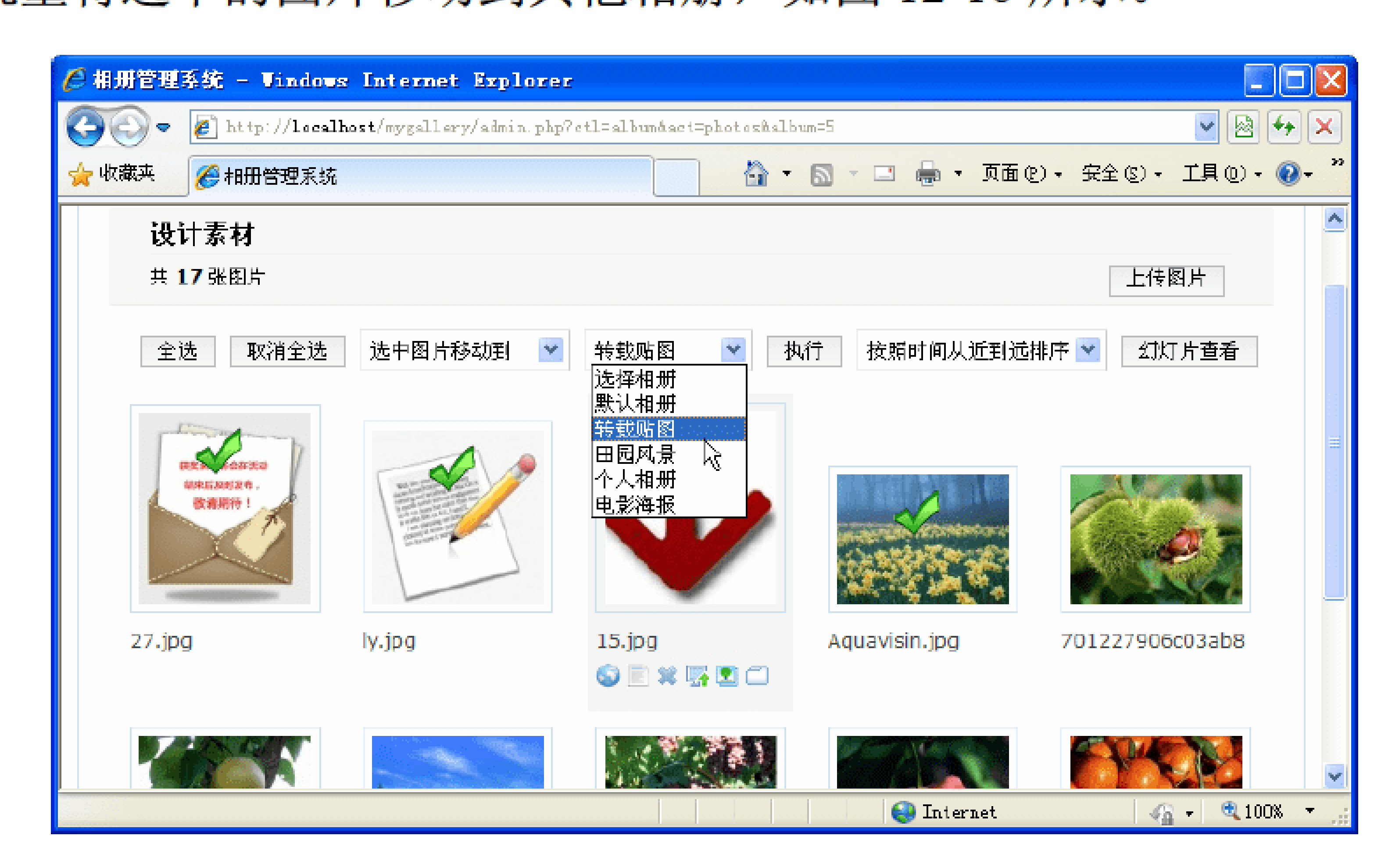


图 12-16　批量移动图片

除此之外单击图片缩略图下方的文件名即可在进入的文本框中对图片进行重命名操作，如图 12-17 所示。

文件名下方的是操作工具栏，其中有 6 个按钮，它们的作用如下。

- 复制图片的 URL 地址，如图 12-18 所示。
- 复制插入图片的 HTML 代码，如图 12-19 所示。
- 删除当前图片。
- 重新上传当前图片。

（2）true
（3）false
（4）array_push
（5）array_combine()
二、选择题
（1）D
（2）D
（3）A
（4）D
（5）B
（6）D
（7）D

第 6 章　字符串处理

一、填空题
（1）\'
（2）数字
（3）strlen($str)
（4）B
（5）www.itzcn.com/
二、选择题
（1）D
（2）D
（3）B
（4）A
（5）C

第 7 章　常用数据处理

一、填空题
（1）按值传递
（2）func_num_args()、func_get_args()
（3）rand(1, 100)
（4）factorial($number-1)
（5）失败
二、选择题
（1）A

（2）C
（3）D
（4）A
（5）A
（6）D

第 8 章　文件和目录处理

一、填空题
（1）filectime()
（2）size
（3）file
（4）index
（5）scandir()
（6）disk_free_space()
二、选择题
（1）A
（2）D
（3）B
（4）D
（5）C
（6）D

第 9 章　与 Web 页面交互

一、填空题
（1）method
（2）<?php echo $song;?>
（3）$_GET["season"]
（4）setcookie("web")
二、选择题
（1）A
（2）A
（3）D
（4）D
（5）C
（6）C
（7）B

第 10 章 MySQL 数据库与 PHP 处理

一、填空题

（1）MYSQL_CLIENT_COMPRESS

（2）mysql_pconnect()

（3）mysql_list_dbs()

（4）mysql_fetch_assoc($result)

（5）type

（6）mysqli_num_rows

二、选择题

（1）B

（2）C

（3）A

（4）D

（5）C

第 11 章 PHP 高级开发

一、填空题

（1）xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");

（2）status

（3）xmlHttp.responseXML

（4）md5("php")

二、选择题

（1）D

（2）A

（3）D

（4）C

（5）D